NTNU
Norwegian University of
Science and Technology

Customer Engagement in Agile Sofware Development

Marianne Worren

Master of Science in Computer Science
Submission date: July 2010
Supervisor: Eric Monteiro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

How can an organization facilitate customer engagement in an agile software development project, emphasizing on the project's needs and constraints?

The objective of this thesis is to answer this questions based on literature and an empirical study of a software development company.

Assignment given: 15. January 2010
Supervisor: Eric Monteiro, IDI

Abstract

Agile methods promise an ideal approach to customer involvement. However, the success relies on having a full-time dedicated, on-site customer representative working in close collaboration with the developers throughout all phases of the project in order to provide the team with ongoing domain expertize. For many projects, providing this form of customer involvement is infeasible. Organizations are therefore left with finding a more viable way of practicing customer involvement

Through a case study of medium-sized, multi-national organization, practicing agile software development with off-site customer representatives, I illuminated the challenges emerging from this situation. By providing a framework for practitioners, I present my suggestions on how to decided on the right customer representative, and what support functions that needs to be established in order for the customer involvement to be successful.

Preface

This thesis is the final part of my degree in Master of Technology in Computer Science at the Norwegian University of Science and Technology.

This thesis is based on my pre-study from autumn 2009, done together with Magnar Wium.

First, I would like to thank my supervisor Eric Monteiro for his guidance and useful comments along the way. I would also like to thank Torgeir Dingsøyr for his help and for providing me with relevant literature.

A special thanks goes to Ole Markus With, for many fruitful discussions and for his great effort in helping me structure this thesis.

Further, I would like to thank all the employees at Sportradar whom I were in contact with during the course of my study, for being so open and helpful and for letting me up close on their work. I am especially grateful to the guys at the Stockholm office, for giving me such a warm welcome. Without their help, the information I base my analysis on would not have been as complete.

As a final acknowledgment I would like to thank Magnar Wium for leaving me with a solid foundation for this thesis.

Trondheim, July 2th, 2010
Marianne Worren

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the innovative and complex world of technology and software development, ambiguous, vague and changing requirements raised by the customer pose a challenge for system developers. More sophisticated technology has made the customer equally more demanding, and to fulfill their high expectations, new ways of managing the customer relationship has to be considered—the waterfall model no longer holds its stand.

As an answer to this, agile software development has risen as a new breed of methodologies. Although the principles agile methods propose are neither new nor radical paradigm shifts in software development (Fitzgerald et al., 2006a), the methods have been manifested and they have gained increasingly attention in the literature the last couple of decades.

Together with eXtreme Programming (XP), Scrum is one of the most popular and frequently used agile methods (Fitzgerald et al., 2006a). Through a flexible, iterative way of working, products are delivered faster and more specialized to the customer.

Agile methods rely on a significant degree of customer involvement, and stress the success factor of engaging the customer in the whole development process. Continuous customer engagement is the ideal situation for all software development project that are dependent on domain knowledge about the customer marked in order to make a satisfying product for the customer. However, for many projects, this is simply not feasible, and organizations are forced to find a more justifiable way of practicing customer involvement, as *resource constrains*–like time pressure, budget, and efficiency requirements–does not allow for the requirements to be followed by the book.

Customer involvement is receiving significant attention in the literature, as

well as in the software development business. In recent years, there have been many success stories of customer involvement in agile methodologies, and there is a general consensus among practitioners and researchers that communication with and feedback from the customer, as well as the potential end user of the system is beneficial. However, the literature is still lacking a precise description and recommendations to *how* to involve the customer.

This thesis presents a theory of agile software development, focusing on customer involvement in a Scrum project.

Given that customer involvement is desirable, the question I would like to answer is this:

> *How is customer involvement practiced in an agile software development team, given the resource constraints in the organization?*

To answer this question I have conducted a case study of a medium-sized software development company, Sportradar, for about eight months. Sportradar coordinate all their projects using Scrum, involving customer representatives throughout the whole development process.

My research–data gathering and analysis of the empirical results–was guided by these research questions:

1. Which mechanisms for customer involvement are applied in the organization where the study took place?

2. What are the trade-offs with having an off-site customer representative?

3. How can an organization improve the effectiveness of the support functions and practices for customer involvement?

These questions will be answered in relation to the analysis of the research results.

1.1 Structure of the Thesis

This thesis consists of the three following parts:

Part I: Theory

This part will equip the reader with relevant literature needed in order to follow the analysis of the case study. The first chapter examines the plan-driven and agile methods literature in general, and Scrum literature in particular. The second chapter discusses customer involvement in general, while the third chapter of this part goes one step further and look at customer involvement in agile software development. Combined, these three theory chapters will provide the reader with the body of knowledge on what has already been done in the field, what is lacking in the research and at the same time identify the importance of this research.

Part II: Empirical Study

I start this section by explaining the research methods and the data gather methods used–interview, observation and document analysis–and justify why I chose these methods, reflecting on both the advantages and shortcomings of applying these methods to this specific case study.

The results from my research are then presented as a more or less chronological history of the project I have been following.

Part III: Analysis and Discussion

In the last part of my thesis I analyze and reflect over the empirical findings. Based on the analysis of my research results, my aim is to answer the research questions provided in this chapter. IBased on the lessons learned from the case and from my literature review, I have comed up with a framework for practitionares, for how to practice customer involvement and who to choose as the customer representative.

I conclude my thesis with a set of implications for practitionares and researchers and with suggestions for further research on the topic of customer engagement in agile software development.

Part I

Theory

Chapter 2

Software Development Methodologies

This chapter provides the relevant literature for the research, beginning with a brief overview of the history of software development, followed by the shift leading us to today's state of the art. The characteristics of agile development will be presented, with a closer look on one of the most popular and commonly used agile methods—Scrum—and see how this method is being practiced.

2.1 Traditional Software Engineering

During the last decades we have seen a shift in the way of practicing software development, from traditional software engineering [1] to a greater use of methodologies with a predefined set of values, principles and practices. But before the introduction of structured, agile methods, there were still some established software best practices, and certain models were followed.

2.1.1 Defining a Method for Software Engineering

The goal of software development projects has always been to deliver a working system that meets requirements of the customer and the user within a given time frame and resources. The process of going from a conceptual and vague idea of some functionality to delivering a working system is truly com-

[1] The term *traditional software engineering/development* is used for the non-agile methodologies, also known as Tayloristic approaches or plan-based processes.

plex. Since the early days, software engineering practitioners and researchers have recognized the need for a method to control this process.

The first methodical approaches to software development were guided by the *life cycle model* (Nerur et al., 2005). The idea was to control risk by breaking software development into manageable activities for sequential execution, each activity building on the previous activity. The classical model when it comes to sequential software development is the *waterfall model*, described by Royce (1970). This model is in many ways the archetype software development method, and gained popularity already in the late 70's (Larman and Basili, 2003).

The waterfall model begins with a complete analysis of user requirements. The analysis is done through months of intense interaction with user and customers. From this, engineers would establish a definitive and exhaustive set of features, functional requirements, and non-functional requirements. This information is well-documented for the next stage—design—where engineers collaborate with others, such as database and data structure experts, to create the optimal architecture for the system. Next, the programmers implement the well-documented design, and finally, the complete, system is tested and shipped.

One could argue that this model presents rather intuitive approach to creating software: *know what you are going to build before you start building it.* The document-driven and plan-driven software approach, which the waterfall model represents, has its roots in other engineering disciplines where functional requirements could be more precisely defined at an early stage and would not be subject to change in later stages in the development. Near to total knowledge are tried to be derived and captured, and a defined processes promise predictability.

However, traditional software development faces at least two challenges:

- Creating software involves a great deal of knowledge transfer in communicating domain knowledge, from customer to developers.
- In most cases the people who commission the building of a software system do not know exactly what they want and are unable to articulate everything that they know (Parnas and Clements, 1986).

The latter implies that requirements will change. A sequential process could not handle this change in requirements very well, since changes in a later phase would mean the all the documentation created in an earlier phase would have to be re-written, causing the initial planning to be in vain.

This problem with strictly plan-driven development has been well known in the software community for decades. Some of the first people to explicitly highlight the issue were Parnas and Clements (1986) stating this: given a complete knowledge of the requirements, it will still be hidden requirements related to the design which only can be revealed through trial and error.

Every software development process since the 1970's has included mechanisms for addressing the problem of changing requirements. A popular subset of these are the *iterative processes*, which are characterized by running the waterfall phases in several loops, having a gradually more complex system prototype as output. Feedback from customers and users is gathered after each cycle to adjust and set the goals for the next system prototype. The perhaps most well known of these are *the spiral model*, described in Boehm (1988). Like the waterfall model, the spiral model base itself on truly up-front requirement analysis, and address the problems with unknown and changing requirements by building a scaled down system prototype in the early project life cycle for stakeholder feedback.

2.1.2 Customer Involvement in Plan-driven Methods

Customer involvement is not something new that came with the agile package, as we will see in chapter 3. Plan-driven methods also make use of customer representatives to provide input and feedback. Due to the good planning artifacts that traditional methods offer, the customer representative does not need to be constantly available, as agile methods requires. However, there is one big challenge with customer involvement in plan-driven methods, and that is to keep project control from falling into the hands of overly bureaucratic contract managers (Boehm and Turner, 2003b).

2.2 Agile Software Development

There is no agreement on what the concept of *agile* actually refers to (Abrahamsson et al., 2002), but one could argue that agility in the context of software development is to *prepare for change and embrace it rather than rejecting it* (Williams and Cockburn, 2003). Most teams use a mixture of different agile methodologies (Davies and Sedley, 2009), so throught this thesis I will refere to this mixture as agile.

Even though the agile methods are a relatively recent addition to the flora of software developing methodologies, several respected software engineers

actually recognized the advantage of having an agile approach to developing systems as far back as the 1950's (Larman and Basili, 2003), and the *iterative enhancement technique* of developing software was introduced in Basili and Turner (1975).

Despite the fact that prominent engineers had success with *iterative* and *incremental* software development, this approach was at large seen as emergency solution when the ideal approach of a predefined, documentation driven process was not an option until the 1990's. It was during this decade agile practitioners started to formally define their methods, an effort which accumulated in the *Agile Software Manifesto* published by a group of software practitioners and consultants in 2001. Their focal values honored by agile practitioners are presented in the four following principles:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

These are the common principles for all agile methods, and what radically makes them differ from traditional methodologies which focus on a predefined processes. I will now explain each of the four principles.

Responding to change over following a plan

Unlike traditional methods, agile does not avoid change and complexity but rather embrace it and account for it (Williams and Cockburn, 2003). To be able to respond to change, agile methods use *short development iterations* to incrementally build the final product. The idea is to use the increased knowledge generated within the short iteration as a guide to set the goals for the next iteration.

Individuals and interactions over processes and tools

Agile development emphasizes the people factor (Cockburn and Highsmith, 2001) and the human role of each individual (Abrahamsson et al., 2002). Each individual is seen an asset and vital part of the team. Physical interaction among people is more valued than any technical communication tool, and team spirit and relationship are important (Abrahamsson et al., 2002).

Focus on people is further combined with *collective effort* in developing software, as the role of the basic developers has shifted from being a craftsmen carrying out plans given from above, to take system wide decisions on architecture and design. The influence and decision power is taken from the management and transferred to the team, as the *software developers and the customer make most of the decisions* (Nerur et al., 2005). As a side note, this collaborative decision making requires the management to give up on some of their previously control and authority established in a plan-driven environment.

Development should be carried out by small teams, with collocated members working closely together. This should help to unite the team, enhance communication between the developers and improve feedback.

Working software over comprehensive documentation

An implication of the fundamental principle of agility is that changing system requirements will leave traditional documentation outdated. Dusty piles of unread, outdated documentation add little value to the customer, and an agile practitioner will therefore minimize the documentation effort, and rather focus on the activity that generates the most value for the customer: running code. Agile focus on the code and thereby aim to reduce overhead (Constantine, 2002).

Customer collaboration over contract negotiation

Agile development requires the developers to collaborate with the customer in an iterative fashion throughout the whole development process; a *strong customer involvement* has proved to be a critical success factor of agile development (Chow and Cao, 2008). Although the degree of involvement might vary, a common characteristic is the use of mechanisms for *customer feedback* to drive the iterations. Especially XP apply an extremely high degree of customer involvement, requiring the customer to be on-site and constantly available for the developers (Koskela and Abrahamsson, 2004).

2.2.1 Team Size

Team size is probably the most important factor to determine if agile is applicable or not (Lindvall et al., 2002). According to the literature, Scrum is most suitable for relatively small development teams. The exact number of recommended participants differ, but a usual recommendation is that there

Figure 2.1: Communication link with three intermediaries and a 5 percent communication loss leads to only 77 percent of the information getting to the developer. The percentage indicates how much of the original information is intact.

should be no more than ten engineers (Abrahamsson et al., 2002; Cockburn, 2007; Boehm and Turner, 2003b). Agile is not applicable if the team size gets too big, as face-to-face communication brakes down and coordinating interfaces becomes an issue (Lindvall et al., 2002). With more than twelve programmers, there is a bigger reliance on tacit knowledge, which is difficult to build without good, close communication (Cockburn, 2007). On the other hand, if the team is too small — less that three — agile will add overhead to the development (Cockburn and Highsmith, 2001).

2.2.2 Communication

Communication is a key success factor in agile methods, as reflected in the agile principle *Individuals and interactions over processes and tools.* Communication can take many forms, each form offering different degrees of effectiveness, where the most effective type is *person-to-person, face-to-face* (Melnik and Maurer, 2004).

Face-to-face interaction is required when facilitating knowledge and information sharing between communities. This will help improve the level of mutual understanding and to develop *social relationships* Hislop (2009).

Without face-to-face interaction between customers and developers, information loss will occurs when communication through a chains of intermediaries (Chau and Maurer, 2004). With a chain of five people, assuming that 5 percent of relevant information is lost in each transfer between each of the stages, nearly a quarter of the information does not reach the developer (see picture 2.1). By removing any intermediaries between the customer and the developer it reduces the information loss drastically: with a communication error of 5 percent, 95 percent of the information now gets to the coder.

To reduce loss of data, knowledge sharing in agile is done trough *short knowledge transfer chains* (Melnik and Maurer, 2004), and the chain is shortened by direct communication and collaboration.

Communication is a crucial factor for an agile team to succeed, as interaction

between individuals is one of the core activities of agile methodologies. Agile teams embrace the power of conversations, and activities like the daily Scrum meetings facilitate knowledge sharing and a collaborative team work. Team planning and *retrospective* (i.e. project review meeting) are activities where knowledge is gathered and shared between people with different experience across the organization (Boehm and Turner, 2003b).

Media richness theory suggests that face-to-face channels offer the richest form of communication due to transmission of multiple cues, like tone of voice and body language. Personal focus, feelings and emotions are part of what is being communicated, and the communicating parties get rapid feedback from each other (Daft and Lengel, 1986).

Knowledge sharing is an important goal of communication. Organizations delivering highly customized products or services, or producing innovative products, invest mainly in person-to-person knowledge sharing (Hansen et al., 1999). The same goes for organizations with boundary spanners, who needs to carry information across departments and resolve the ambiguity about goal, issues and course of actions (Daft and Lengel, 1986).

The *richness of information* says something about the ability of information to enable understanding and overcome different perspectives (Daft and Lengel, 1986). Transfer of rich transformation can go through different *communication media*, where the richest communication is by face-to-face, because of immediate feedback, transmission of cues, personalization and language variety. Studies have also showed that richer media support quicker decision making. Audio and video communication is also considered as high richness mediums (Dennis and Kinney, 1998).

Media of low richness, like documents and e-mail, process fewer cues and restrict feedback, but never the less, it is offers the best media when it comes to effectively transferring of standard data and well understood messages. However, it is worth mentioning that Daft and Lengel (1986) are controversial with their suggestion for which medias should prove most efficient in what situations; Evidence of rich media's ability to improve the performance of uncertain task has not been very supportive (Dennis and Valacich, 1999). Face-to-face communication is not necessarily the richest medium for communication, as the "best" medium depends on for example the importance of immediate feedback in a given situation. A conversation may very well be biased by too much cues and personalization.

2.2.3 Stakeholder Involvement

As in traditional requirements engineering, agile methods rely on a significant degree of *stakeholder involvement* for gathering of requirements — though in a slightly more idealized picture (Paetsch et al., 2003). Agile methods are people-centric, and so it is essential that the stakeholders are actively involved (Nerur and Balijepally, 2007; Pikkarainen et al., 2008).

The notion of stakeholders in agile literature is often restricted to the customers. However, a broader notion has been brought to light, and developers and users are often included in the stakeholder definition. On the far end of the spectrum, stakeholders can include everyone in the organization, from sales and marketing, sponsors and resource management to all employees at the client company as well as the users (Conboy et al., 2009). everyone who is affected by the outcome of the project or all the individuals or organizations that have a say in the development, should be identified as stakeholders (Eberlein and do Prado Leite, 2002).

2.2.4 Scrum

Scrum was defined by Schwaber et al. (1995) as an approach for managing the software development process. Schwaber states that *changing variables*, such as user requirements, makes the software development process impossible to define. He thinks that the best way to control chaos is to be flexible and responsive when change happen.

The Scrum characteristics is that only the first phase (i.e. initial planning) and the last phase (i.e. closure) of the process are considered as defined, since the inputs and outputs of these actives are well-defined. The flexible part of the method is the iterative *Sprints* where the product is incrementally built. Whenever tactic knowledge and experience is available, this is used to build the software; otherwise, trial and error is used. The Sprints are considered empirical processes. Controls, such as risk management, are put in each iteration to avoid chaos while still remaining flexible.

Even though Scrum lays out a set of predefined roles and describes the development process (e.g. iterations and stand up meetings), it leaves open for the developers to choose their own software development techniques, methods, and practices for the implementation process (Abrahamsson et al., 2003), as opposed to XP, which lays out twelve principles for the implementation process (e.g. pair-programming, open workspace and on-site customer) (Beck, 1999).

Roles in Scrum

Scrum identifies six roles that have different tasks and purposes during the process and its practices (Abrahamsson et al., 2002). Bearing in mind agile method's focus on empowering the people, offering a set of predefined roles might seem contradictory, since each person is being put into a role with pre-defined tasks and responsibilities. But these roles are not as final as the literature may put it. The team members are not confined to a specialized role, which enables them to self-organize (Nerur et al., 2005). In agile you have to build the roles around people, and not the other way around[2].

Notice that Scrum does not define a *project manager* in the same way as in traditional development approach, where there is one person responsible for the decision making. In agile development, the project manager is no longer a planner and controller, but a facilitator. The product owner, though responsible for the backlog, is neither fully in charge of the decision making. Both roles have less authority as they would have had in a traditional development method, as decision making in agile is done in a collaborative fashion (Nerur et al., 2005; Karlström and Runeson, 2006).

The following definitions of the six Scrum roles are drawn from Abrahamsson et al. (2002).

Scum master The Scrum master is responsible for the project being carried out according to Scrum practices. Their role is not that of a traditional project leader, but more as a coach. Talking to the development team, management and the customer, the Scrum master is responsible for removing impediment so that the project team can work as productively as possible.

Product owner The product owner is selected by the Scrum master, the customer and the management. The product owner is officially responsible for the project, including the creation of the product backlog (see process description next), and steering the project in the right direction through the incremental Sprints. This is done by prioritizing and selecting functionality for implementation in front of each Sprint, as well as constantly refining the backlog using the Sprint reviews as primary input for that activity.

Scrum team The Scrum development team is responsible for delivering the product and to do the actual implementation. The decision of *how* to

[2]http://www.threeriversinstitute.org/blog/?p=503

achieve the goal of each Sprint is left up to the project team.

Management The management is in charge of selecting the product owner, measuring the process and reducing the backlog.

Customer The customer participates in making the product backlog. They work closely together with the developers, so that they are well-informed and competent to consider possible adjustments that emerge during the life-cycle of the development.

Scrum Phases

Schwaber et al. (1995) defines Scum as having the three following phases, which makes up the process:

Pre-Game Phase This phase consists of planning and architectural design and defining the system. This is done in close collaboration with the customer. The tacit manifestation of the first conceptual dentition of the system is the *product backlog list*, or *backlog* for short. This list initially contains high level requirements and goals. The architectural design is based on the product backlog. The output for this activity should be a high level design for the backlog, supporting what is currently known about the system.

The development phase The development phase, known as the *Sprint*, is the agile part of the method. The Sprint iteratively builds functionality to the product, refining it as knowledge of the final product emerges. Each Sprint usually last between 2-4 weeks, depending on the complexity of the system, risk assessment and oversight desired. The goal of each Sprint is to create running software that can be presented to the project stakeholders.

Each Sprint starts off with a planning meeting organized by the Scrum master. This meeting has two parts. The first part of the meeting includes the development team, product owner and often customer and management representatives. The purpose is to choose goals and functionality from the backlog to be included into the product. The second part only includes the development team, and the goal is to identify how the product increment is implemented during the Sprint.

The primary managerial control tool during the day-to-day developing is the *daily Scrum meetings* held by the Scrum master. These are usually held

each morning before the programmers start to work, and are organized to continuously keep track of the working progress. What has been done since last meeting is summed up, and what should be done until the next one, is identified. The goal is also to identify any impediments in the process, and find out how to remove them.

The last day of the Sprint, a *Sprint review meeting* is held. This meeting includes a demonstration of the results of the Sprint by the Scrum master and development team to the management, customers and product owner. Following the demonstration, discussions between the participants are held, which can lead to new backlog items, or refinement of existing items.

Post-game phase It is a managerial decision when to enter the last stage, which will lead to a product release. The decision is based on where the project is standing in terms of development variables, like user requirements and time. This phase includes the traditional software engineering activities related to software release: integration, system test and creation of user documentation. Once these activities are fulfilled, the product is finally released.

Chapter 3

Customer Involvement

In traditional software engineering, the customer or the user were normally just involved in the first phase of the development process only, during planning and making of the requirements. After the specifications had been made and the requirements document had been handed over to the technical team, the customer barely interacted with the developers (Parnas and Clements, 1986).

In 1999 the system development community came to agree on a set of best practices of human-centered design in the development process. International standards for *human-centered design* processes was clearly defined and published as an ISO standard, ISO 13407, an authoritative statement that incorporate human factors with the objective of enhancing effectiveness and efficiency (Bevan, 2001).

ISO 13407 represents a maturing of user involvement, not merely in the first phase of the project, but in all phases of the system design. This ISO standard lifted up the models of *user centered design* (UCD), which follows this ISO standard.

Having recognized user involvement as something valuable, the question is no longer *if* we should do it, but *how*.

3.1 Approaches for Customer Involvement

As we touched on, customer involvement is common practice in software development but the degree of engagement vary. This is implies that a precise approach for involvement is difficult to define. However, the following list

provided by Kujala (2003) provides a rough classification of the different types of approaches:

- Participatory design
- Contextual design
- Ethnographic studies
- User centered design

Participatory design relates to the *participative customer role*, where the user is involved by analyzing the organizational requirements.

In *contextual design* the idea is to study the work process and to describe and redesign them by changing role structure, supporting task, automating and elimination unnecessary steps.

In *ethnographic studies* the effort to understand work processes is taken one step further, by letting the system designers plunging into the natural environment of the user over an extended period of time. The goal is to fully understand the needs and challenges of potential users. This insight is drawn by applying research methods coming form social anthropology such as interviews and observation.

User centered design (UCD) is an involvement approach that is characterized by early focus on users and tasks, empirical measurement in terms of usefulness and iterative design. I will now give a deeper description of this approach, as UCD is a popular topic in the literature, and is closely related to customer involvement.

3.1.1 User Centered Design and Usability Engineering

As the name suggest, UCD focus on user involvement in software design, and has been described and glorified as "an adjunct to an effective agile process, the missing link back to users that can turn an incomplete process into one that can reliably deliver good solutions for users, not just good code" (Constantine, 2002). The goal of user centered design is the development of useful and usable products, filling the need and requirements of the user.

There is no clear process defining UCD (Kujala, 2003), but a general UCD approach can roughly be divided into three phases (Detweiler, 2007):

- Phase 1: Understanding the user by creating written user profiles through extensive research directly involving the user through interviews and focus group.
- Phase 2: Defining interaction by evaluating the documents from phase 1 and creating use cases based on the previous identified users.
- Phase 3: Designing the user interface based on the use cases from phase 2 by iteratively building and evaluating a system user interface (UI) prototype against real users.

While just briefly touching the key steps within the traditional UCD-process, it is can be said that UCD requires an excessive degree of user involvement. These steps do not fit with every project. The fact that UCD in its pure form can be quite expensive has been pointed out in the IS literature by among others Nielsen (1994), who humorously highlights this issue:

> "When asking how many usability specialists it takes to change a light bulb, the answer might well be four: Two to conduct a field study and task analysis to determine whether people really need light, one to observe the user who actually screws in the light bulb, and one to control the video camera filming the event. It is certainly true that one should study user needs before implementing supposed solutions to those problems. Even so, the perception that anybody touching usability will come down with a bad case of budget overruns is keeping many software projects from achieving the level of usability their users deserve."

As a remedy for this problem of doing UCD, Nielsen (1994) proposes a light weight alternative to UCD called *discount usability*. The discount usability engineering[1] method is based on the use of the three following techniques:

- *Scenarios*: simple prototypes that address a small subset of functionality in each user feedback iteration.
- *Simplified thinking aloud*: a light weight technique that is often used in large scale usability engineering, where the user talks aloud their thoughts and doubts while using a prototype. The result is videotaped and analyzed by interaction designers and psychologists.

[1]Usability engineering is a set of techniques and method used within UCD to achieve the goals of UCD.

- *Heuristic evaluation*: testing the interface against usability rules and principals.

3.2 Boundary Spanning

When a company is faced with having to manage the interaction and collaboration between people with a limited amount of common knowledge, value and identity, we say that the company is dealing with a *cross community, boundary*[2] *spanning knowledge process* (Hislop, 2009). With increased globalization and collaboration among different disciplines, these types of knowledge management is becoming more and more common.

The performance of boundary spanning has different definitions. Levina and Vaast (2005) refer to it as the practice of successfully engaging engineers and marketing specialists to relate the practices of their professional fields. It can thus be thought of as a specific way of relationship management.

Both individuals and artifacts can perform boundary spanning. Individuals are usually referred to as *boundary spanners*, while artifacts are referred to as *boundary objects*.

As with boundary spanning as an *activity*, there are many definitions in the literature on boundary spanner *individuals*. Levina and Vaast (2005) characterize boundary spanners as "vital individuals who facilitate the sharing of expertise by linking two or more groups of people separated by location, hierarchy or function". Brown and Duguid (1998) go one step further in their definition, by splitting the boundary spanning role into brokers and translators. *Brokers* participate in both communities, while *translators* are affected by the consequences of the message they carry, rather than any community. Tushman and Scaland (1981a) define boundary spanning individuals as someone who are *internal stars*, meaning they are frequently consulted on work related matters, while at the same time gathering and transferring information from outside their subunit.

A boundary spanner has multiple responsibilities, and one of them is being responsible for information exchange between the organization and its task environment (Leifer and Delbecq, 1978). For example, they can have a bridging function between the IT experts and the user of the system (Damodaran,

[2]Boundary in this context is by Leifer and Delbecq (1978) defined as "the demarcation line or region between one system and another, that protects the members of the system from extra-systemic indulgences and that regulates the flow of information, material and people into or out of the system".

1996). Thus, their work tasks require high competence in at least two fields of practices. At the same time they should act professional and objective, and expertise and discretion are vital requirements (Aldrich and Herker., 1977). To be able to gain respect from the developers, their status is based on perceived technical competence (Tushman and Scaland, 1981a).

Being a boundary spanner is associated with challenges and a heavy burden for most people. The boundary spanner must have competence in multiple knowledge domains which often have conflicting interests, something that can lead to stress and burnout (Jagdip Singh and Rhoads, 1994). Another source of overburden will be of all information in the environment should require immediate attention, something that can happen since boundary spanners are expected to serve as the organizational defense against information overload (Aldrich and Herker., 1977), because a boundary spanner is also expected to act as an information filter. Finding a suitable, competent person willing to take on this challenging role can be a difficult and time requiring process, but it is critical for the boundary spanning activity to be successful.

On the positive side there are also advantages associated with having a boundary spanning role. First off all, because they are a critical resource for the organization they will have informational power and status (Tushman and Scaland, 1981b). Their role is usually recognized as important, and they gain respect as they easily become an expert on a domain few others in the company are familiar with.

3.3 The Benefits of Customer Involvement

Examples of benefits of customer involvement are not hard to find. Kujala (2003) points to increased sales, increased user productivity, decreased training costs, and decreased user support. Additional benefits include a more accurate assessment of user requirements, prevention of costly, unwanted features and greater user acceptance and understanding (Robey and Farrow, 1982). Customers possess knowledge from a different domain then technicians, and the customers often raise questions and come with requirements that the development would have else never thought of (Kujala, 2003). Thus, user involvement is most efficient in the early stages of the development, when negotiating the requirements.

Feedback is also considered valuable in the later stages of development, as agile methods welcome changing requirement. Cooperating with the developers also has a highly motivating effect on the customer. Study has showed

that customers are passionate about their role on development projects using agile methods (Martin, 2009). Having the customer on the development team will keep them constantly updated on how the product is taking form, and at the same time they will be able to influence the direction the development is going. Research has showed that customers tend to be more satisfied with the product when they have been a part of the development process (Kujala, 2003).

From the developer's side, close customer collaboration has showed to make the developers feel comfortable and more personal responsible for the customers, and thus appreciating a direct and informative dialog with the customer (Hanssen and Fægri, 2006).

If the customer representative role can take be taken on by a boundary spanner, the advantages with customer involvement is even greater. In product development, combining expertise from different domains requires new thinking and a different way of managing knowledge, which can be a new source of innovation (Carlile, 2002; Aldrich and Herker., 1977; Keil and Carmel, 1995), which, again can lead to competitive advantage. Innovativeness requires a lot of information gathering (Leifer and Delbecq, 1978), and this is where the boundary spanner comes in. If a company successfully manages to span its boundaries, the company can derive new knowledge which can be a key organizational advantage (Carlile, 2004).

Chapter 4

Customer Involvement in Agile Development

Agile development and ISO 13407 pulls the software community towards embracing customer involvement and people over technology. As opposed to traditional software development, agile methods emphasize personal communication between customers and developers throughout all phases of the development cycle (Melnik and Maurer, 2004).

As mentioned in the introduction of this chapter, ISO 13407 is a defined standard of best practices of human-centered design in the development process. ISO 13407 lists four principles, where one of them is: "the active involvement of users and a clear understanding of user and task requirements" (Earthy et al., 2001). This maps well to how agile methods emphasize user involvement and knowledge sharing with the developers. ISO 13407 and agile methodology agree on the importance of user involvement.

ISO 13407 is not the only standard for user centered design – A comprehensive range of international standard has been made the last two decades (Bevan, 2001). One advantage with these standards is that they can provide guidance for inexperienced companies, as well as being a useful source of reference. Common for most of these standards is that they often specify general principle rather than precise details on exactly how to practice customer involvement (Bevan, 2001). Therefore, these standards may be regarded by some as challenging constraints (Earthy et al., 2001).

Customer collaboration means that all stakeholders, including customers, users and developers, work together on the same team (Highsmith and Cockburn, 2001). Scrum and XP embrace the customer, treat them as a valuable asset and a contributor, and fully integrate them in the development team.

As we have seen in the two previous chapters, the main challenge when building software is meeting the customer's requirements. Thirty years experimenting with development processes and customer involvement points to that iterative development has proved to be an efficient tool when it comes to dealing with changing requirements.

The agile principles seams to reflect precisely these experiences when it comes to defining a software development process: iterative, customer feedback driven development based on direct communication. The change to agility has proved successful for many companies when it comes to customer satisfaction (Mann and Maurer, 2005; Hansson et al., 2004; Miller, 2005). Would this imply that the IS and software engineering community has finally cracked the code?

Research has showed that it might not be that easy. The various agile methods differ in terms how the fundamental principle of customer involvement is explicitly manifested as requirements for conduction. eXtreme programming (XP) provide one of the most explicitly pronounced guidelines for customer involvement, requiring an on site customer representative to drive the daily development. The Scrum process defines its requirements much looser. In Scrum, the customer should be involved in creating the initial backlog items together with the development team, and contribute by feedback and elaboration on the backlog thorough out the game phase (Abrahamsson et al., 2002). Scrum does not give any real guidance on how and what should be the tasks of the customer representative.

4.1 Who is the Customer?

A customer as in its traditional meaning is the one paying for the product. In agile literature, the customer is a person representing the system users (Nerur et al., 2005). The customer is a requirement expert, a domain expert and a usage expert[1], who can contribute to the backlog with their knowledge about what the user requires. A common characteristic about customer representatives is that they normally possess a business background (Martin et al., 2009); they have few or no technical skills. The responsibilities of the customer are to drive the project and to provide requirements and quality control (Martin, 2009).

The "customer" in agile literature is often a surrogate for the end customer[2]

[1] These are terms applied to the customer in Cockburn (2007)

[2] I will use the term *end customer* on the true customer of the software company, so as

or end user, and often we talk about a *customer representative*. A *customer proxy* is a type of customer representative, who is someone from the development organization who represents the customer. A customer proxy is often an alternative when collaborating with an on-site customer is impossible (Stevenson and Pols, 2004). The customer—proxy or not—has to know the people they are representing and need to have a clear understanding of their needs and requirements. Hence, the terms *usage expert*, *requirement expert* and *domain expert* are also common in the literature. The customer can also put on more than one role, making the definition of a customer even broader and more complex.

As a side note, agile methods have from its introduction grown towards are larger way of thinking about "the customer", now recommending whole *customer teams* (Conboy et al., 2009). However, most of the agile literature still talks about the singe customer, and it is not unusual that the one product owner is the only customer representative (Pikkarainen et al., 2008).

Some literature makes a distinction between the three terms customer *involvement*, *participation* and *engagement*. Muller and Kuhn (1993), for example, suggest that "participation" represent a higher goal than "involvement"[3]. Also, the literature is neither distinct in its use of the terms *user* involvement versus *customer* involvement.

The success of agile projects relies on finding an appropriate customer representative who will be committed and actively participate in the development process (Nerur et al., 2005). Agile literature requires this person to be a *user expert* (Cockburn and Highsmith, 2001), meaning that they should be representative for the customer, knowledgeable and posses the domain knowledge needed by the developers. Further, they should also be authorized, so that they can make his own decisions and not cause delays seeking approval (Boehm and Turner, 2003a). According to XP, the customer should in addition to this be constantly available, be on-site and provide ongoing expertise (Cockburn, 2007).

The customer role can be rather complex and dynamic, as responsibilities differ from project to project and through each stage of the development process. With a high degree of involvement the customer is fully committed and physically available to the developers on a full-time basis. At the other extreme, the customer can be located in another country, contributing strictly at specific, planned time slots trough e-mail, phone or video conferences.

---

to not confuse this person with the customer *representative*.

[3]I, however, will use these terms as synonyms throughout this thesis

However, having a high degree of involvement is only symbolic if the customer does not have any real influence on the decisions being made. How much influence the user representative has can be placed somewhere on a scale reaching from informative involvement, through consultative to participative (Damodaran, 1996):

- **Informative involvement** means that the user only provides and receives information.
- **Consultative involvement** allows users to comment on a predefined service.
- **Participative involvement** requires that the user have influence on the decisions relating to the whole system. This is the highest and most desirable degree of influence a customer can have.

To date, discussions around agile methods have mostly focused on bringing the customer to the developers. The concept of having the developers traveling to the customer's area, to observe and talk to the customer on a day-to-day basis—as agile require the customer to do—has yet to be addressed (Conboy et al., 2009).

4.2 Different Styles of Customer Involvement

The degree and nature of customer involvement comes in all kinds of flavors. The customer can be engaged as an *advisor*, as seen in the longitudinal study by Hanssen and Fægri (2006) of a Norwegian software product company. This company had prior to the study experienced problems with handling changing customer requirements with a waterfall development process, and decided to engage stakeholders directly using an agile method based on stakeholder driven one week development cycles.

The company introduced a project management team consisting of representatives from *sales and marketing*, in order to find the right stakeholders for involvement and manage the customer-developer relationship. The customer representatives were chosen on basis of their perceived domain expertise and willingness to contribute. Their actual responsibility was to evaluate the product after each cycle, and provide feedback that would be used in planning next cycle. The developers considered the customers as valuable advisors, but were sensitive to the fact that *the customer representatives only*

represented a subset of the total customer base, and would therefore make priorisation based on broader market view.

The role of a customer representative can also be more extensive. The customer can be given the main responsibility of developing new product concepts, and engage in the demanding role as *proxy* between the different stakeholders and at the same time being an on site resources for the developers. This kind of customer involvement is typical for extreme programming (Martin et al., 2004).

On the other side of the scale is engagement based on little direct contribution (Detweiler, 2007; Miller, 2005). Hansson et al. (2004) presents a study of a small provider of off-the-self software that focused on getting the end users' feedback quickly into running code. By combing short (daily) agile development cycles and their extensive database of user data from their support service they managed to sustain as a major player in the market they operated in. The users and customers of their service were also engaged in workshops and courses held by company representatives, but neither the customers nor the users had any direct influence over the development process.

This short review of different engagements styles brings out the various choices companies are faced with when planning customer engagement. References made in this section come from studies of good experiences with agile customer involvement. The general positive effect of short cycles with close, often face-to-face, customer communication can been seen as the creation of an efficient channel for knowledge transfer between customers and developers. In Hanssen and Fægri (2006) the developers felt increased motivation and job satisfaction due constant reassuring from the customer that they we building the right thing. A qualitative manifestation of this positive effect of agile development is decreased over time (Mann and Maurer, 2005; Hanssen and Fægri, 2006).

4.3 Challenges

There are many positive extended effects of getting the customer close on the development and focusing on communication over passing documents. However, agile does neither explain exactly *what* customers should do, nor exactly *how* they should do it (Martin et al., 2009). Research on agile development has revealed some challenges companies practicing agile customer engagement has dealt with:

- Managing the customer-developer collaboration

- Engaging the user as well as the customer
- Balancing and supporting the customer representative

Facilitating and managing communication between technical developers and non-technical customers are usually a challenge with customer involvement. I will not go deeper into communication in this chapter, as it is already explained in 2.2.2.

I will use the case studies by Hanssen and Fægri (2006) and Martin et al. (2004) to illustrate the rest of these challenges.

4.3.1 Managing the Customer-Developer Collaboration

In the longitudinal case study by Hanssen and Fægri (2006) they followed the positive effects of increased customer engagement, but also identified some potential problems with this approach. The customer feedback became a significant driving force behind the development. Since the customer representatives were eager to contribute with feedback and development goals, this proved an effective development approaches until the representatives lost interest due to various reasons. This created a problematic void from the developer's point of view, and made them have to take important decisions without proper feedback. The team responsible for managing the customer-developer relationship failed in this situation, and the authors emphasizes in light of this example the importance of keeping close watch on the evolving collaboration between developers and customers.

Further, Hanssen and Fægri (2006) describe how the development team often got a too short time frame to write the sufficient amount off test code. Rather, the development cycles that pushed functionality was prioritized, since those had large perceived value for the customer representatives. This introduced a lot of error in the code.

A general challenge when placing a *non-technical customer representative* in a position of direct collaboration with developers is *balancing the requests from the customer, and at the same the focus on testing, scalable code and other non-visible qualities of the software being built.* This is important when engaging customer regardless of methodology, but can it can be argued that this is somewhat more a crucial challenge in agile development. For XP and Scrum, each agile cycle is a complete product life cycle, and should include both unit testing and proper code review (Abrahamsson et al., 2002). With stakeholder pressure this can become quite a challenge for the development

team, as learned from Hanssen and Fægri (2006). In the case of a non-technical customer being placed as vital part of the development cycle, it is important that they are familiar with the process and their role in this situation.

4.3.2 Engaging the User as Well as the Customer

In Hanssen and Fægri (2006) there are a relatively few customers involved (one per every 3-4 developer), and their requirements sometimes conflicted with those the company perceived as important based on their knowledge of the market. This highlight an important, but problematic aspect of agile customer involvement: As the agile principles calls for removing intermediaries between the stakeholders and developers, a single voice can in many cases get huge impact on what is being developed. There is also no guarantee that the voice of end users is heard. If the selected customer representative is an expert on the case, they might not recognize potential problems that first-time users may have. Further, different users may have very different mental models about the area being addressed by the software, and that may require a broader pool of users providing feedback to recognize potential usability concerns. This can become a problem in the case of few or single representatives (Kane, 2003).

On the positive side, there is evidence of successful integration of broad user feedback as the primary driving force within an agile process, as seen in the case study by Miller (2005). Companies as SAP and Alias had aligned their agile development with parallel UCD-activities, and by using *interactions designers* as proxies for the development team, Alias managed to include many different voices as the designers work against a large spectrum of potential end users. Short development cycles helped turn user needs quickly into running code.

The customer involvement style at Alias can be classified as passive: The customer, or as in this case, *potential users*, can not directly affect what is being developed since all feedback is routed through the interaction designers. This might be sufficient for large and experienced organization as Alias, but the risks of such practice is directly tied how well the selected users represents the total population of users and how well the developers carry out the prioritization process (Hansson et al., 2004).

4.3.3 Balancing and Supporting the Customer Representative

In their study of the customer role in extreme programming, Martin et al. (2004) found that this role can be quite demanding. They studied both on-site and partly off-site customers and discovered that in addition to explicit responsibilities in relation to acceptance testing, requirements generation and quality ensuring, the customer had several implicit responsibilities as a result of their role as organizational proxies between project founders and developer. It is therefore vital to support the new role of the customer representative in software development with additional personnel.

Part II

Empirical Study

Chapter 5

Research Methods

The empirical evidence, which will be presented in chapter 7, is drawn from a case study at Sportradar AG (from here on referred to as Sportradar), a multinational software development company. The research was done from August 2009 to May 2010. During this period I had free access to the Sportradar Trondheim office, and was welcomed to drop by and leave as I pleased.

This thesis is a continuation on a research project conducted through fall 2009, which I wrote together with Magnar Wium. Together we spent approximately one day each week at the Sportradar office in Trondheim. During the spring of 2010 I wrote this master thesis, spending roughly the same amount of time at Sportradar's offices in Trondheim. In addition, I spent one working week at Sportradar's office in Stockholm, Sweden.

Prior to the study, Wium and I had been working as summer interns at Sportradar, so this is how we got in contact with the company. Both worked for two months in 2008, and Wium worked an additional period of two months in 2009 as well as being a part-time employee during fall 2009.

In this study I have focused on the communication links between the end customer, the customer representative and the developers in the context of Scrum. I have been following a project called *the Statistics project*, which is the fourth rebuilding of Sportradar's original Statistics solution (i.e. the *old Statistics*). The Statistics is a distributed project where the developers are located in Trondheim, Norway, the product owner is located in Gera, Germany and sales are located in Stockholm, Sweden.

Through investigation of how individuals on both side of the communication channels acted and how they felt about the collaboration, I tried to get an

overview of what was working, why it was working, what could be improved and how it could be improved. I did this analysis by comparing the literature with the situation at Sportradar. I iterated between maintaining and applying literature to my research throughout the period of my study.

I do not aim to give a complete description or a comprehensive evaluation of how the communication flow work at Sportradar or how it is being managed. This would require more knowledge about upper management and insight of the organization as a whole, which I did not have access to or time to investigate during my research.

5.1 Philosophical Paradigm

A paradigm is a set of shared assumptions or ways of thinking about the world. Different philosophical paradigms implies different views about the nature of our world and the ways we can acquire knowledge about it (Oates, 2006). Underlying assumptions will be influencing how the researchers choose and apply methods as well as how he or she will interpret the results. Thus, in relation to Information Software (IS) research, explicitly stating the research fundament in terms of philosophical paradigm, is a premise for assessing the quality of the result.

The different philosophical assumptions can be classified as positivist, critical or interpretive (Klein and Myers, 1999):

A **positivist** approach implies two basic assumptions: the world is ordered and regular (as opposed to random), and secondly, we can investigate the world objectively (Oates, 2006). The positivist researcher will first come up with a hypothesis. The researcher then collects evidence which in turn are elevated to evidence that unmistakable will reject or confirm the hypothesis. When confirmed, the hypothesis will form a positivist point of view and stand as a universal truth as long as contradictory evidence do not challenge it.

A **critical** researcher assumes that there exists alienating conditions that can be identified and removed in order to enhance the opportunities for realizing human potential within the research context (Klein and Myers, 1999).

IS research is classified as **interpretive** if it is assumed that our knowledge of reality is gained only through social constructions such as language, consciousness, shared meanings, documents, tools and other artifacts (Klein and Myers, 1999). Contrary to positivist researchers, interpretive researchers do not consider themselves independent of the phenomena under study, but

believe that their action of presence has influence on the findings.

In this thesis I haven taken the latter perspective, namely an interpretive approach. I try to gain an understanding of how a methodological theory is applied in a real life setting, and how groups of individuals perceives its value. I also try to identify whether or not the applied theory, like Scrum, can been seen as beneficial for the company in its current form. Given this context I believe the knowledge I'm searching for in this thesis should be gained by interacting with and observing the context for customer engagement in agile software development. I do this in order to extract my thesis, thus keeping an interpretive mindset as the philosophical fundament for my research.

In the process of gaining insight I believe that the researcher's background and assumptions strongly influence how they conduct and interpret the research. I am aware of the fact that my presence within the context I am studying might influence the participants and thereby color the results.

5.2 The Different Research Methods

Interpretive research can make use of either qualitative or quantitative research methods:

Quantitative research methods has traditionally been applied in positivist research, where statistical analysis of numbers has been used to draw meaning form samples generated by methods as surveys and closed laboratory experiments.

Qualitative research methods has it roots from natural science, and are characterized by how numbers serves as strong scientific evidence. The goal is often not to derive precise theories but to create understanding of a phenomena by gaining rich insight into a phenomena entangled within a social context. As the phenomena under study is inseparable for the social context it exist in, qualitative research usually requires the researcher to interact directly within context using methods such as interviews and observation as tools for gaining insight.

The research questions in this thesis are aimed at creating an understanding of how agile methods in relation to customer engagement changes the way software is created, and what challenges the people involved in this process are faced with. This is undoubtedly a complex topic closely tied to the socio-technical process of developing software, and that is why I find it appropriate to make use of the qualitative research approach.

Further, I have been using a research method know as a *case study*. A case study is interpretative research strategy that has been describes as:

> "[..] an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident." (Oates, 2006)

Case study is appropriate for answering my research questions, as it allows me to gather rich, qualitative data by observing a phenomena within the software industry.

5.3 Data Collection

I chose to conduct a case study focusing on one specific project within a software development company. The reason I decided to analyze one project was because I wanted to get an in-depth knowledge of the history and situation of the project. Also, I would not have been able to visit merely the Stockholm office I were to study other projects as well, as these project have sales directors located in other parts of Europe, and have a different customer segment.

I used three data gathering methods for generating (mostly) qualitative data to base my analysis on:

- **Interviews**

  More than 30 semi-structured in-depth interviews

  E-mail communication

- **Observation**

  Being present in two meetings with an end customer

  Being present in video conferences (con-calls)

  Being present in several Scrum meetings

  Engaging in informal discussions (e.g. during lunch and by the coffee machine)

- **Document analysis**

  Reading and evaluating documents from an online database, such as the Sprint backlog and the requirement specifications

I will now go into each of these three methods and reflect on how they have been applied in this specific research, as well as discuss their drawbacks and advantages.

Interviews

Interview has been the primary research method. The interviews performed falls under the category of **semi-structured interviews**: I prepared certain themes formulated as questions, while always leaving room for improvisation. The interviews lasted from 10 minutes to 120 minutes, depending on the themes I wanted to cover and the schedule of the participants.

Most of the interviews were held at Sportradar's Trondheim department during a period of eight months, from the beginning of September 2009 until late May 2010, with a break in December 2009.

During fall 2009, most of the interviews were performed together with my research partner, Wium. spring 2010 I continued the study alone. My interview targets were employees of Sportradar working on the Statistics project, as well as the management, sales directors and an end customer.

I always carried a small *note book* with me. I took notes during con call-meetings, meetings with the customers and during all interviews. I wrote down quotes, words, following-up questions that came into mind, either for the person I was interviewing or someone else (so that I did not have to interrupt the subject while they were speaking, but could pick up the thread afterwards) and reactions and communicated feelings of subject (e.g. surprised, nodding, reluctant, sighing et cetera). I also drew maps of communication flow, some of which are redrawn on the computer and included in the text of the case study. The notes were then written out on a computer as soon as I got the opportunity, while I still had the conversations fresh in mind.

In addition to always taking notes, most of the interviews during fall 2009 were recorded, but only some where transcribed due to limited time. I did not transcribe the whole interviews, but the quotations I found interesting. In spring 2010 I only recorded a few of the interviews, as I wanted to keep the in interview situations more relaxed. Also, as the amount of data was growing, transcribing every interview was infeasible due to the time constraints I was working under.

In Trondheim, the spoken language was Norwegian. The recordings and notes were then translated into English by me. With the sales directors in

Stockholm, the conversations were mostly held in English, while some of the informal discussions switched over to Swedish. With the Swedish customer, I conducted the interview in Swedish, and then translated the quotes into English.

At the turn of the year, my focus slightly shifted, from following the involvement of product owner PO1 to following the involvement of a new graphic designer, sales and the true end customer. This shift in focus was a natural consequence of the change in the course of the project; from constantly implementing new features based on PO1's requests, the project went over in a phase where no new features where added. Rather, bugs were being fixed and the user interface was redesigned. Because of this new phase and the new role of PO1, who gained a less influential role, I choose to follow the communication and collaboration between the developers and other stakeholder–the end customer, sales, the graphical and interaction designer–who all got more involved in the project.

In May 2010 I spent one week at Sportradar's office in Stockholm, where I got to interview and get to know the four employees located in Stockholm: two sales directors, one graphical designers and one project manager. I also got to meet one of the biggest client of the Statistics solution in Scandinavia, which I will call Sportnews. I were able to visit Sportnews in relation to two meetings with sales director SD1 and employees from Sportnews. On one of these meetings, I interviewed the web manager at Sportnews, responsible for the integration of the Statistics solution at the client web page.

The number of team members on the Statistics varied throughout my study. The Statistics project had over the course of my research a total of twelve members on the Scrum team, in addition to stakeholders in the management, sales department and among end customers.

The following table presents the team members and stakeholders I got to interview:

Note the difference between a *product* manager and a *project* manager. As oppose to the technical project manager PM1 for the Statistics, the product manager M2 at Sportradar's Stockholm office had more administrative tasks, and were in close dialog with the sales directors and upper management. The project manager PM1 is the Scrum master of the Statistics project, but I will use the term project manager throughout the empirical research, as that is his official title and what the developers refers to him as.

The number of interviews listed in Table 5.1 is an underestimation, as I have only counted for the number of *formal* interviews, meaning the times I made

Table 5.1: Number of interviews performed with each of the subjects.

Group	Role	Coding	Location	#
Management	Manager	M1	Trondheim	1
	Project manager	PM1	Trondheim	3
	Product manager	M2	Stockholm	4
	Product owner	PO1	Gera	2
Sales	Sales Director	SD1	Stockholm	5
	Sales Director	SD2	Stockholm	1
	Sales Director	SD3	Stockholm	2
Developers	Developer	D1	Trondheim	1
	Developer	D2	Trondheim	3
	Developer	D3	Trondheim	2
	System architect	SA1	Trondheim	4
	Graphic designer	GD1	Trondheim	3
	Graphic designer	GD2	Stockholm	3
Customer	Web Admin in Sportnews	C1	Stockholm	1

an agreement with the subject, sat down and asked them prepared questions and took notes and/or taped the interview. All the casual conversations with the participants where I used the opportunity to ask them questions are not counted for, as it is hard to keep track of the exact numbers of these conversations. I got to know the participants quite well, and conversations often started when I asked someone (usually PM1, SA1, SD1 or M2) for clarifications, and they would start explaining more in detail or escalating the answer into a conversation, sometimes lasting for more than one hour. A significant amount of the data provided in my case study is learned by these conversations.

Management, sales and end customer were naturally those who required an extra effort to get in touch with. For example, Wium and I first contacted PO1 through upper management in Trondheim, who informed the Gera department about the research we were conducting. After some initial e-mail communication were Wium and I presented our self and our research topic, an agreement was made to meet with PO1 on a visit he was planning to Trondheim in late September. We spoke to him again a couple of weeks later on another trip he took to Trondheim.

As I was limited to interview PO1 when he was visiting Trondheim, I have not been able to get his insights directly on themes that have emerged in later stages. At the turn of the year I followed his activity in the project management tool at Sportradar, but I have had no direct contact with PO1.

I have had no contact with PO1's supervisor either - one of the top management employees in Sportradar — as Wium and I were told initially by the technical director in Trondheim that we should avoid contacting him due to his busy schedule. Due to this, PO1's supervisor is seldom mentioned in the case. From a Trondheim perspective, he had no active role in the project, but had rather let PO1 be his representative. However, PO1's supervisor had in the planning phase of the new Statistics had a more defined role in the project, and was then being referred to as the project leader by sales and management.

The remaining nine team members were located in Trondheim, filling the roles of six developers, one part-time graphical designer, one system architect and one project manager. I conducted formal interviews with five of them.

Participant Observation

Observation is a field research method useful for finding out what people are doing, and not only what they say they are doing. I have been a participant observer in the sense that I have experienced the situation at Sportradar from the view points of the developers working there. During the research period, I have attended several meetings: morning meetings, Sprint reviews, Sprint planning and general management meetings. The management meetings and review meetings were so called *con-call meetings*, were the Gera representatives and other managers from different offices around Europe participated through phone and web camera. I took notes during the meetings, using pen and paper, and wrote it out on my laptop directly after the meetings, when what had been said and done were still fresh in mind.

As I spent about one day each week at Sportradar's Trondheim office I also conducted what can be classified as **general observation**. I were sitting in the middle of the developers' working area, an open office space, and engaged in informal discussions were I presented my theories and asked for opinions and insights. I did not make use of note taking during this observation, as it would have seemed unnatural and may lessened the developers comfort and willingness to speak openly and freely. I was also a participant observer on one of my meetings with the end customer in Sweden, and also after the meeting when I got a tour around the client's office, where the client's software developers were working.

Document Analysis

Document analysis was, together with interviews, essential to get a clear picture of the development of the project *prior* to my research. The case study tells the whole history of the project, even from before my research started. To get insight on what had happened before, analyzing documents was an important part of the data gathering.

I were given full access to the project management tool used by Sportradar. This allowed me to get an indication on what had been done in each Sprint. I could study how long each feature took to implement, how long time it was estimated to take, and who was working on it. This was just rough indicators, as many of the developers were open-handed with registration their work in the management tool as well as features sometimes got shifted relatively to when they actually got finished. The management tool was still useful in the sense that it could help identify incidences were a feature took significantly more effort than what it originally had been estimated to. More important, the management tool made me able to read the features description created by the product owner PO1, as I got access to various documentation created by PO1 in relation to his early conceptual research. During the redesigning phase of the project in 2010, I could also observe how the role of PO1 changed, from the degree of details and the nature of the tasks he posted. I could also access reports from all Sprint reviews, helping me to get an understanding of the progress of the project.

5.4 Evaluation of Research Quality

The challenge of assessing the quality of interpretative field research has been addressed by Klein and Myers (1999). I will now evaluate me research in relation to the seven principles suggested by Klein and Myers (1999).

I: The fundamental principle of the hermeneutic circle

The first principle suggests that all human understanding is achieved by iterating between the interdependent meaning of *parts* and of the *whole* they from.

Even though my case study evolved around one project only, I actively sought to identify its place in the lager organizational frame. I loosely compared the Statistics project with other ongoing projects at Sportradar, to make it easier

to point out what was special and different about the Statistics. The difference was also pointed out by the management from Trondheim and some of the developers. As my understanding and insight of Sportradar and the development of the Statistics project grew, I started to get the whole picture of the story. Having insight into a larger part of the company structure I found tremendously important, in order to understand the actions of individuals and their consequences. I constantly measured these events against my acquired insight, given them meaning in relation to a larger context.

II: The principle of contextualization

The second principle emphasize the importance of *critical reflection on the social and historical background* of the research setting, so that the intended audience can see how the current situation under investigation emerged.

This principle is important to reflect over in relation to this research for two main reasons. Firstly, I have been following the Statistics from September 2009 to May 2010, while the project history, from the old Statistics solution to this fourth rebuilding, goes several years back. Background information is essential for understanding the current status and the process of developing the Statistics.

Secondly, the background and tradition of the company itself will also play an important part when it comes to how and why the customer engagement is carried out in its current form. For example, the resemblance of agile development at the Trondheim office even *before* Scrum was officially introduce, is important to be aware of, as this affected the decisions Sportradar took in choosing what agile principles to adapt to. I also says something about the decisions that were *not* being made, due to the comfort in the established environment.

I interviewed two other product owners in the company, working on other projects than the Statistics, which are not directly included in the case study. I did this before the scope of this thesis got narrowed down to following just one project. These reflections has not found its way into the thesis, but have served as useful exercise for data gathering and in order to get a bigger picture on the organization.

Further, I have reflected on the social, cultural and historical background of the various individuals playing a part in the development of the Statistics. For example, being aware of the differences between the German and Scandinavian culture, both the professional and socially, has helped me understand the developers perception the two segments had of each other. If I

should have gone deeper into my research, I would have tried to get an even deeper understanding of the cultural issues and what effect these have on the collaboration.

The principle of contextualization can be related to the principle of the hermeneutic circle: If the researcher has an understanding of the whole picture, it is easier to put the research results into context. In my example, I had to be careful not to get a tunnel view on the project I was following, but rather watch the context around. What other projects were going on in the same open office location in Trondheim and how they were all being managed, helped me see the whole picture and made it easier for me to understand what was going on when I zoomed in on the Statistics project.

III: The principle of interaction

As a third principle Klein and Myers (1999) states that the interaction between the researchers and their subjects require critical reflection on how the research materials and data were socially constructed.

When I interpreted and analyzed the data, I tried to be aware of how my age, gender, educational background, programming experience and so on had bungled my interaction with the subjects. By preparing open ended questions, letting the subject speak out and mirroring the language of the informant rather than using my own terminology, I tried not to let my nature and personality have too much influence on the conversation. However, I realize that to some extend I still affected the answers of the informants in one way or the other. Some of the factors might even have worked for my advantage, as I experienced that the interaction with the participants in my study was strongly affected by my previous working experience as a summer intern in 2007. Whenever I mentioned my background as a programmer at Sportradar, I felt I immediately gained some sort of trust and respects from the subjects. Without this background, I doubt I would have been able to gather the detailed amount of data — most of which is not present in this thesis, but still useful for giving me an understanding of the situation, and to base my analysis upon.

Wium and I went into the interview with the product owner being concerned about of how lack of trust might influence the outcome or limit our chances for further interaction with him and the Gera side of the Sportradar organization. I perceived that lack of trust might come from the social and cultural distance between us, as well as the potential of him perceiving Wium and me as part of the Trondheim department and thereby limit his willingness to communicate

any critical views about that department.

As the interviews with the product owner were some of the first interviews I performed, I were not yet comfortable with pushing the subject for more details, and I did not question his answers during the interview. Even if I missed out on some of the details, I think that this initial effort to bridge the gap between us helped in creating a good social environment. In the end, the closure of the interviews opened for further interviews, and this might not have happened if I had pushed on more sensitive topics.

Through my interaction with the Trondheim department, and especially the Stockholm office, I believe that I raised the awareness about the customer's role in the development process. I believe that some of the developers initially had taken the current ordering for granted, and given little thought in relation to end users representation within in the development process. As I were interacting with Trondheim as researchers for about eight months, my early data will not in the same way as the latter data reflect developers more critical stance in terms of the process. I have taken this into account when I have analyzed the data.

At the Stockholm office I got direct, explicit feedback on my research; the sales directors and the manager were positive towards my research and several times pointed out the importance of someone sheding some light on the topic at Sportradar.

IV: The principle of abstraction and generalization

The fourth principle emphasize the importance of describing the link between particular instances the researchers has encountered through the empirical work and the theoretical lens the meanings of these has been understood against.

In the analysis of the data I have to some degree generalized the case study at Sportradar, in order to make the results and conclusion yield for other companies in similar situations, and not just for Sportradar. To do this abstraction, I have tried to draw the links of similarities between cases in my literature study and my case study, and I have applied terms from the literature I see fit into the situation at Sportradar. For example, when I discovered the many different roles of the product owner at the Statistics, I draw a link to the literature, whenever the product owner's role coincide with those of the literature.

V: The principle of dialogical reasoning

Interpretive research requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and the actual findings. This requires the researcher to explicitly inform the audience how their assumptions have change over time.

This thesis started out as a study of communication between actors in global software development, but as we soon discovered, Sportradar proved a suitable context for learning about the practice of customer involvement. Agile methodologies also seemed to be a hot topic in the literature, that required more research in the field of customer involvement.

VI: The principle of multiple interpretations

The researcher should be sensitive to possible differences in interpretations among the participants, typically expressed in multiply stories of the same sequence of events (Klein and Myers, 1999).

I have been studying a phenomena which involves physically and culturally separated participants, who daily interact with each other. Thus, they participate in the same sequence of events, most likely to be seen from different viewpoints. One safe way to go about this situation is to make sure both sides are heard and represented in the results. I have therefore tried to gain insight in both Gera, Trondheim and Stockholm, to discover possible different vies and aspects of the development process and specific events. My visit to the office in Stockholm were mostly done because I wanted to uncover the customer side of the project, with sales' and the customer's view of the product, the process and the Trondheim office. My goal of gaining a holistic view of the situation was to a large extend fulfilled. I felt I gained a balanced picture of what had happened during the history of the Statistics project and what was going on in the present.

Although the field research for the case study was quite in depth, more quantitative validation and detailed statistics would have made it more likely to uncover multiple interpretation, and hence strengthen the results of the analysis. For example, counting the number of mails between the different participants of this study would have provided me with solid proof of the extend of the communication, and made me able to draw a more correct communication map of the organization. However, that would have been an intrusive way of gathering data, and the benefits of doing so would possibly not worth the effort, as I feel I uncovered a sufficient part of the communication picture.

In those instances where I found conflicting views, I have explicitly stated this, as well as my reflection on why these differences come into play.

VII: The principle of suspicion

The last principle emphasize that the researcher need to look out for possible false preconceptions that certain actors or the whole organization is communicating or assuming.

My perception of opposing viewpoints among the informants can be related to the principle of suspicion. One example of this is how the German department in many ways came across as a black box - not just to the researchers, but to the developers in Trondheim as well. The exact number of employees located in Germany and the division of labor in relation to the Statistics projects was not clear, and none of the developers in Trondheim seamed to know exactly what was going on there. Visiting the German department to uncover the "rumors" related to this office would therefore have been a priority if I had had more time.

In some of my interviews I experienced that the subject acted like they were reluctance towards opening up and giving away sensitive information. This might happen when the researcher's questions comes across as accusations towards the company or the subject, causing a protective attitude. This is an undesirable situation, and to avoid this kind of defensive behavior, it is important that the researcher make the objective of the research clear from the beginning. The interviewee has to be convinced that the researcher is not an intruder, trying to dig up flaws from their own working place. The researcher must try to come across as someone who can be trusted with the given information, and who will use it in a constructive manner. However, there is not always much time to establish this kind of thrust before an interview, something I experienced when I was interviewing one of Sportradar's customers. Prior to the interview, I had e-mail him a description of my research, explicitly stating that "my aim is to help Sportradar improve their product development". I was not given much time to explain this further during the allocated time for the interview, so the customer still might have felt suspicious towards me. The customer did not say much negative about the management of the project, but I have to be aware that the answers he gave might have been out of politeness to the Sportradar sales director who was present.

The principle of suspicion mostly applied when I was interviewing stakeholders outside of Trondheim, but for most of the interviews, I felt I was given

very frank and honest answers. Especially sales were clearly positive to my research, and they found my topic both interesting and useful for Sportradar, and therefore were willing to contribute with important information in order for my research to work in a constructive manner for Sportradar. The same goes for the management in Trondheim. The developers, who knew me quite well and seamed comfortable around me, also provided my research with open and honest input.

Chapter 6

Sportradar AG

Sportradar AG is the company where the case study took place. It is a group of smaller companies, consisting of roughly 200 employees divided into seven departments in six European countries. The company delivers various sports related services; Services including analytical monitoring of the betting market in order to identify and prevent match-fixing in football, as well as various sport statistic services to bookmakers and the media market worldwide.

6.1 Company History

The Sportradar AG company history dates back to year 2000 when two graduates from the Norwegian University of Science and Technology (NTNU) formed a software start-up named Market Monitor. The company was then specializing in so called *high accuracy crawling technology*. This technology was developed and used for crawling through online betting sites and detecting instances were two bookmakers had odds for a single sport event that deviate so much that no matter what the outcome of the match was, the better would win. These instances were causing bookmarks much headache, and Market Monitor received venture capital from a major player in the bookmaker market to develop the technology further.

In 2002 Market Monitor joined forces with Sportinformation Europe (ISE). ISE is a German based company with large networks of scouts [1] and various other means for collection sports data. This partnership allowed Market

[1] a scout in the sport industry is someone watching games in different sports live and plotting data about the match close to real time to the happenings

6.2 Organizational Structure

Figure 6.1: The organizational strucure of Sportradar AG

Monitor to developed a new line of products delivering statistics and match fixtures for bookmakers. In the years that followed Market Monitor became engaged in several partnerships. This made the company extended their product suit to include, among others, fraud detection systems used for rigged sporting events and statistic services targeting media. In 2008 Market monitor extended its partner group further and at the same time changed name to Sportradar AG.

Sportradar AG has to day the world's largest betting database and delivers services to more than 200 customers in more than 40 countries.

The Sportradar AG group has offices in Switzerland, Germany, Sweden, Estonia, Austria and the Czech republic. The responsibilities of the different branches span from marketing, legal advice, sales to data collection, product development and software implementation. We will now look in more detailed on the Trondheim and Gera departments, since they are the primary actors in relation to the development of the new statistics-project.

6.2.1 Trondheim

Trondheim hold Sportradar AG's Norwegian branch: Sportradar AS [2]. This is also Sportradar's primary technical department with about 40 employees, close to all of them working as software developers. The Trondheim department has also some responsibilities in terms of technical support. The man-

[2]From here on referred to as Sportradar

agement in Trondheim consists of two people, following up on the projects and keeping a dialog with the rest of the company, like the Gera office and the Stockholm office.

6.2.2 Gera

The Gera office is the German branch of the Sportradar group. The Gera office is responsible for adding sports information into the database, a database that serves as the foundation for most of Sportradar's products. It is vital that fresh sport information find its way into the database. This is partly carried out by software based on computer vision algorithms, and partly by Sportradar personnel manually reading information of television sports cast and typing this information into the database through interfaces created by the Trondheim developers. Gera is also doing quality assurance in terms of monitoring the Sportradar's online statistics services.

6.2.3 Sales

Sportradar has employed a group of sales agents. Some of these agents are located at various Sportradar offices around Europe, but for the most part these agents work as autonomous units.

6.2.4 The Stockholm Office

Three of Sportradar's sales agents are located in Sportradar's office in Stockholm, Sweeden. This office consisted of a product manager, a graphical designer (since October 2009) and the sales agents. As some of Sportradar's customers are located in Stockholm, this is a natural place for having a sales office. The office is equipped with a meeting room, used when inviting customers and potential customers to discuss the Sportradar products. Often, the sales agents visit the customer at their location, as the customers usually have a very busy schedule. The sales directors and the manager at the Stockholm office also use a majority of their working days traveling abroad, attending exhibitions and meeting with off-site customers.

6.2.5 The Customer Segment

Sportradar operate in a complex environment, with both media clients and bookmakers as customers. In the betting industry there are a few major players, and when they decide on a solution, competitors will follow. Market leaders are lucrative customers and of big economic values for companies like Sportradar. Thus, it is important to satisfy the requirements from these customers.

6.3 The Statistics Project

One of Sportradar's core products is an online sport statistics service, presented as a web page. The Statistics product is a web application on top of a data base containing sport related information. It provides a wide range of sport data, supporting 32 different sports. The web page lets the user browse information such as current status of a specific tournament, that be the Olympic Games, Champions League, Norwegian Tippeligian, the world cup in football and so on. The user of the system can look up the teams playing each other, goals scored, next planed matches, number of yellow cards given to a specific player, et cetera.

The web application is created and maintained by Sportradar's technical department in Trondheim.

The customers of the Statistics services are primarily bookmakers such as *bwin* and *betway*. Lately, more and more media clients have become customers of this service. The media clients are mostly Internet based newspapers, such as Swedish *Aftonbladet*, Spanish *MARCA.COM* and German *Sport1*.

The stakeholders of the Statistics project are everyone in Sportradar AG who has a direct interest in the outcome of the product:

- The sales directors selling the Statistics product.
- The managers in Trondheim and Gera, as well as upper management of the organization.

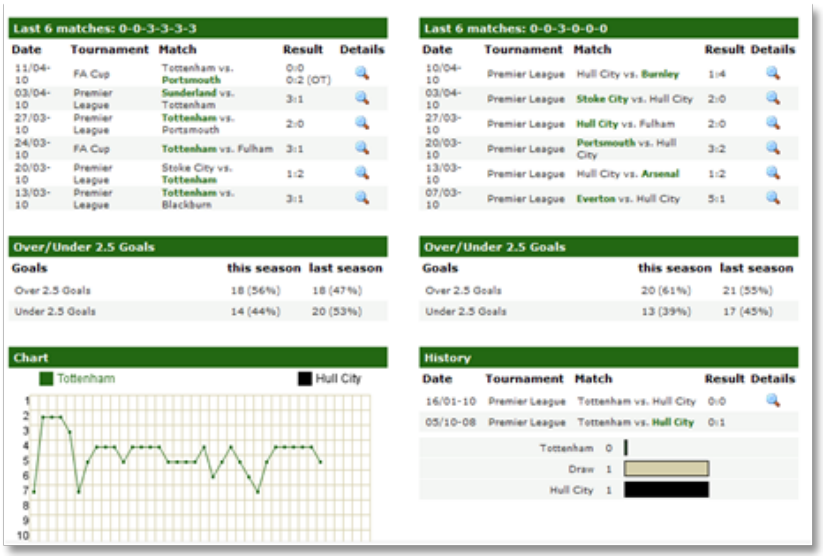

Figure 6.2: A sample from the old Statistics solution, showing a comparison of two teams.

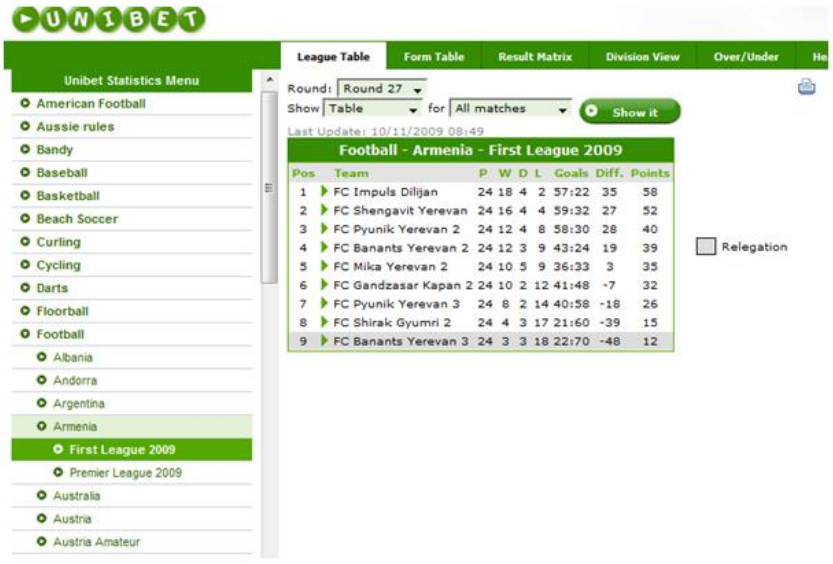

Figure 6.3: A sample from the old Statistics solution, showing a league table.

- The current customers of the Statistics solution.

These are the main stakeholders of the Statistics; Some are more included in the development process, others have no degree of participation.

6.4 Tools for Communication and Project Management

Sportradar use several tools to support their development process. This process is dependent on the collaboration with various partners in the organization. This section describes the most important tools in terms of facilitating the collaboration, as we will refer to them in later parts of this thesis.

6.4.1 Redmine

Redmine is an open source project management tool with a web based interface and a rich set of features[3]. Redmine was introduced aligned with the Trondheim department's introduction of Scrum, and is used by the developers primary to assess feature descriptions, enter spent time of each feature and break down features to smaller units of work. The management in Trondheim use the tool to track project status as well as allocated resources. Stakeholders outside of Trondheim, like the various product owners and sales directors, use Redmine to monitor progress as well as adding new feature descriptions. Redmine was at the time of this research the primary collaboration tool among the departments in Sportradar. It has also been agreed upon that all written communication shall go through the mail-function in Redmine.

The sales directors are encouraged to register issues in Redmine whenever their customers ask for something out of the ordinary, like features Sportradar do not offer in their products. The sales directors do not need to log into Redmine, but simply send an e-mail to a certain e-mail address that directs the message right into Redmine. The issue then get sorted and labeled by either the project managers or the managers in Trondheim. For more "serious" issues, or questions about the products, by the sales directors redirect the customers to the support center .

[3]See www.redmine.org for more information

6.4.2 WebEx

WebEx is a video conference tool that allows participants to share and view documents and applications in real time. The largest Sportradar departments have meetings rooms with conference call equipment ans computers running the WebEx software on large screen. WebEx conference calls, referred to as *con-calls* in the organization, are used weekly by upper management in Sportradar to discuss important issues in plenum. The Sprint review meetings of the Statistics are also held as such sessions, which includes the product owner, his supervisor, the project manager and (usually) the system architect.

The smaller offices run a simple web camera with a head set microphone.

6.4.3 Medialogin

Medialogin is a tool for the media customers to handle the statistical data they have bought from Sportradar, including functionality to adopt their own options. It allows the customers to, among other things, download sport statistics, manage their banners at their webpage as well as the layout of the features. Medialogin is also used by the sales directors to monitor data about their customers, like what XML feeds the customers have been delivered and when it was delivered. Medialogin is developed by Sportradar on request from sales in Sytockholm, as a way of keeping track of the services delivered to each customer.

6.5 Development Process

For Sportradar's established products, additional features and fixes gets requested by sales, and sometimes customers, through e-mail or phone communication with the Trondheim office. This is part of the technical support service Trondheim provides. Two-three developers are responsible for taking request and passing them on to the developer who are most experienced with the issue. If the requests are perceived by Trondheim as quick fixes, they get implemented straight way. On the other hand, if the requests will require extensive planing and a more demanding coding efforts, the request gets put on hold and discussed in the weekly management meeting. During this meeting various proposed features and fixes are discussed and if the management agree, the new features are allocated resource and time.

For new products, a more elaborate development process is being used. For the organization, an important criteria for the allocation of resources for new projects is a sound division of labor between the development department in Trondheim and the rest of the group. Shared responsibility between implementing and deciding what to implement between Trondheim and the various other departments has been incorporated into Sportradar's use of Scrum. The Trondheim department started experimenting with Scrum in 2008, and the methodology has been applied for all major development efforts since early 2009.

Chapter 7

Case: The Statistics Project

This chapter follows the history of one of Sportradar's biggest projects, called *the Statistics project*. The project involves many stakeholders, ranging from customers to sales directors to managers. The developers and the product owner have been working on improving a fourth version of the Statistics project, from fall 2008 to summer 2010.

This chapter starts off with the background history of the Statistics, pointing out the shortcomings with the old solution. This is to explain why there was a need for throwing out the old Statistics—for the fourth time since 2004—and replace it by a completely new solution. The next three sections tells the whole history of the development process of the new Statistics, from phase 1: the planning of the initial requirements, to phase 2: the implementation, and phase 3: redesigning the user interface towards the release date in June 2010.

7.1 Background (2004–2009)

The main revenue generated by the Statistics solution comes from Sportradar customizing the service for each client. This is done by taking a subset of the total Statistics, and then change the layout according to the client's web page layout. The client can either access their version of the Statistics from Sportradar's web server—a so called *hosted solution*—or they can get the data as XML feed, delivered to their own web servers.

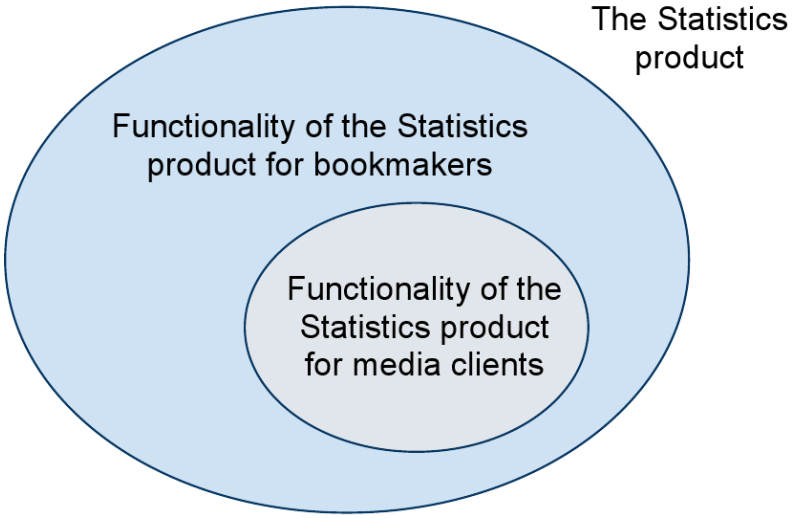

Figure 7.1: The media customers requires less functionality than the bookmakers.

The customers of the Statistics service have traditionally been bookmaker sites who wants to include the Statistics into their own site, so that the end user can use the sports data an as reference when planning bets on a match. The last couple of years, more and more online newspapers—*media clients*—have become customers of the same service, and this has created some challenges. Few newspapers are interested in the entire spectrum of sports related data. Media clients often have very specific requirements of what data they want and how they want it to be displayed. This required the programmers in Trondheim to do much configuration for each client in a system that was not originally designed to support this kind of flexibility. This problem was most visible in Trondheim, were the programmers daily had to struggle with configuration issues.

> "Bookmakers are more nerdy—They just requires the raw data. Media clients, on the other hand, just want to see the most important [information]."
> — *Sales director SD1*

The Statistics product held stand for about four years, of which the media market had been more or less standing still. However, in today's world where technology evolves in a rapid speed, keeping up with ones competitors becomes increasingly important, in this was showed in the customer's requests.

Figure 7.2: Example of functionality from the old Statistics

Sportradar's sales agents around Europe had for the last few years noticed an increasing amount of difficulties in meeting the demands of their clients. Customers were complaining about the functionality, and requested a new solution.

> "Key information about the matches were missing. [..] There is an increasing need for usability in the market. We need a smarter navigation system. [..] The translation was poor. There was absolutely a strong need for a new solution."
> — *Customer C1*

The pressure from the sales department accumulated in a decision from upper management to allocate resources to meet the challenges related to the old Statistics. From the Trondheim office's point of view these requests posed two main problems:

Firstly, as we touch on earlier, the code which the Statistics was based on was not easy to change or extend. Creating a new interface was almost impossible because it did not exist any separation of the presentation code and business logic. Because of this it would be very difficult to do something novel, without rewriting the entire architecture.

Secondly, Trondheim had troubles getting an overview of exactly how and what to include in terms of functionality and new features.

> "They say they want something 'fancy', but what is 'fancy'? It can be anything."
> — *Project manager PM1*

If resources for development should be allocated, it was agreement in Trondheim for a completely new system to be created.

Since the Gera department was involved in the old Statistics, this office was the natural choice for stakeholder representation. The Gera department got the task of analyzing the limitations of the old system and to come up with a new and improved concept. The development and implementation would mainly be done in Trondheim, but before the developers could start working, a technical demonstration had to be approved by sales and upper management at a company gathering that was to be held in Prague in May 2009.

Stakeholders of the statistics can be identified as the developers, the customers, management in Sportradar, sales and product owner.

Prior to the new Statistics, at least two other projects had been developed in Trondheim using Scrum with product owners outside of Trondheim. Having successful experience with this, the same recipe of structure was also chosen for the new Statistics, as we will go deeper into in section 7.3.1.

7.2 Planning of the Initial Requirements (October 2008–May 2009)

Gera hired a former sport journalist, customer and user of the old Statistics solution to work on the concept for the new Statistics product. From a Scrum perspective, this person was the *product owner*, PO1. From the developers side, PO1 was though of as "the customer", and acted as the *customer representative*[1] on the team. PO1 started working for Sportradar in October 2008, and together with his supervisor, they were the head of the Gera department.

PO1's market research for the new concept included collaboration with various sales people and clients of Sportradar (mostly done thorough e-mail and telephone communication), parallel with studying various online statistics providers and reading literature on web usability and interaction design.

Planning the initial requirement took PO1 and his supervisor about five months and the result was presented as a flash-based prototype of the user interface, a detailed user interface design and a comprehensive document–about 200 pages–describing the limitations of the old statistics and suggestions for features that would satisfy both the media market as well as the bookmaker market.

This is a sample from the document that illustrates its level of details:

[1] As PO1 was in-house, he was also what the agile literature calls a *customer proxy*.

> "If you now want to get to the English Premier League, you can roll the mouse over the section 'Europe' in the local navigation. A mouse over function opens up. In this window all national leagues Sportradar covers are displayed in section 'Federations'. These federations are presented by using the international short cut defined by the International Olympic Committee (IOC)."

This is another sample from the same document, explaining an important feature–deep linking–related to client configuration:

> "IV.3.Deep Linking In chapter I.2 we have described the need for a solution that offers the possibility to link directly to a certain page. This is important not only for a number of features we suggest to implement, but also for media clients who focus on certain features. E.g. a media client might want to present a goal scorer list of the English Premier League or the Order of Play of the next day at Wimbledon to his users. He needs to have the option to directly link to the page deep within our data world."

Deep linking came to be one of the most important requirements for the new Statistics, as this was a feature the customers often felt was missing in the old Statistics, according to the sales directors and manager in Stockholm. Project manager PM2 described deep linking as a service that offers a link generator for a specific page. For example, if a newspaper want to write about a specific game at their web page, then they should be able to provide a link in the text that points directly to this information, and not just to the landing page of the Statistics, which had been the case with the old solution. Or, if sales in Stockholm got a request from a customer saying they wanted a direct link to the historical data of all football matches between Liverpool and Arsenal F.C., then the developers in Trondheim should be able to simply add the required arguments to the URL and send the link to the customer. When the users click the link, they should be redirected to a page where the tab containing the requested information is already highlighted.

However, this was not the definition that got communicated to the developers.

> "The developers were told at a meeting with the project manager that deep linking was to insert a subset of components from the Statistics into the customer's web page. [..] Deep linking

is not a technical term. But maybe it's a common term in the sports media industry? I don't know."
— *Developer D2*

The misunderstandings of what deep linking actually referred to led to a great deal of misunderstandings during implementation.

Work on the Statistics project began in Trondheim in early April 2009, when PO1 came to Trondheim to create the initial product backlog. His visit lasted for only one day. PO1 first got a short introduction to Redmine before he used the rest of the day to create about forty features in Redmine. PO1 also talked briefly with some of the developers in Trondheim, but spent the majority of his time working alone in Redmine, after having discussed the old Statistics with SA1. No one of the other developers interacted with PO1.

"I was sitting here with [SA1] and we went through our current Statistics and looking through it, side by side, and created building block by building block."
— *Product owner PO1*

In the following weeks PO1 kept adding features to the backlog by accessing Redmine from Germany. In parallel to this, Trondheim had by May 1 2009 hired two student who were going to work for the summer–D1 and D2–, and hired a project manager PM1 who started working on the *technical* proof of concept. The proof of concept had to be finished before the company meeting in Prague in the end of May 2009.

In addition to the feature descriptions in Redmine, the project had at this point a rich documentation stack: a 190 pages product specification, an interaction model and layout specifications for the application design. PO1 intended the documentation to speak for itself, so to say, and assumed that it would be read by the developers, as a way of saving time.

The creation of the documents was never fully communicated to the developers. The documents were available in Redmine, but not directly tied to the shorter feature descriptions which the developers used. One result from this was that the developer who was assigned the task of implementing the new navigation system –a summer intern– did not study the additional specifications in the document. He spent some time implementing based on the old navigation solution before he learned that his solution would not be sufficient. At this point he contacted the project manager PM1, who clarified the requirements by e-mail communication with PO1.

7.2.1 The Company Meeting in Prague (May 2009)

The meeting in Prague was perceived as a formality by the developers in Trondheim. They were more concerned with getting a framework up and running, so that those students working for the summer who were to arrive in June 2009 would have relevant tasks to work on. The backlog was not considered at this point, but the developers felt that they had a decent understanding of what they should build.

> "We did not focus on features, but rather to display something that was flexible. [Creating such a system] has been done in this office before."
> — *Project manager PM1*

The proof of concept got presented at the company meeting in end of May 2009 for sales and marketing and upper management. The management was positive to the concept and approved it. However, even though no one objected to the concept, the sales personnel in Stockholm felt they did not have the chance to come with any critique. SD1 explained this by saying that PO1 had been working about half a year on the concept, the company meeting was more of a presentation of PO1's work. The sales directors did not get to see any clickable version of the concept, so there was no specific features to complain on. The meeting was not so much an arena for discussion and rearrangement of the requirements as it was a presentation.

> "There wasn't any room for feedback on the concept."
> — *Product Manager M2*

After the demonstration of the proof of concept, sales were asked by the upper management to come up with the *top three prioritized requirements*, based on the requests often posed by the customers. Deep linking was such an obvious and underlying requirements, that sales did not have to include this in their list. The three next most important priorities were presented in a one-page document (it was specifically stated from the upper management th),at the document should not exceed one page) which were handed over to PO1.

Pointed out by the sellers as one of the top requirements was the importance of the systems being flexible, meaning that the customers would be able to pick out the features they wanted and tailor and integrate it into their own web page. This was pointed out several times by all of the sales directors.

> "The key selling point is that the system is flexible [..] Configuration is important for our customers."
> — *Sales director SD2*

Naturally, the sales directors were concerned about selling points like features that would make the product stand out from the competitors. This concerned was based on experience from customers who had made some complains, saying "my table look just like this competitor, and I find it everywhere". Questions concerning the *client setup* and *client integration* were therefore often posed by the customers to the sales directors.

7.3 Implementing Features (June–November 2009)

After the company gathering in Prague, the development effort picked up in Trondheim. The system was to be rebuilt more or less from scratch, so the management gave the developers freedom to explore with the architecture and design. By being detached from the old Statistics, the developers could easily take independent design decisions and there was room for experimenting with new technology. The only fixed dates were that 80 percent of the product should be finished on the 1st of September 2009, and that the beta testing should begin sometime before Christmas 2009.

7.3.1 The Development Process

At Sportradar there was a natural environment for introducing agile methods, as the culture resembled an agile environment even long before Scrum was formally introduced in 2009. Sportradar have had customer representative on other projects before product owner PO1 attended the Statistics, even though not to the same degree and with the same agile formalities; Sportradar also had the advantage of having an open office space; The developers were not used to write all specifications up front, and they did not use documentation. These three factors made it easier to introduce Scrum, as less change in company culture were needed. In fact, introducing a plan driven methodology would in many ways have been much more of an effort than introducing Scrum.

The Sprints were planed as two week development efforts. There was initially focus on implementation work only, but unit testing also became part of the work in later stages of the development.

Sprint Planning Meetings

Each Sprint got kicked off with a full day of planning. Prior to each meeting the project manager PM1 would receive an e-mail form the product owner PO1 with requests for the upcoming Sprint. The team used a projector to run Redmine on a big screen. In the first part of the meeting PM1 would walk through the backlog of features in Redmine and open the ones he had got requested by PO1, so that the team could watch the full description and discuss, based on the descriptions, whether it was feasible or not to include. Unless the team had strong objections, the feature got placed under the upcoming Sprint backlog, which was also represented in Redmine.

During the second part of the meeting, PO1 appointed the tasks to the developers, who estimate how long each feature would take to implement.

Sprint Reviews

The Sprints were ended with a phone meeting between the project manager PM1 in Trondheim and the product owner PO1 and his supervisor in Gera. The system architect SA1 would normally also be present at the Trondheim side. The Sprint review meeting was basically a walk-through of what had been implemented in the Sprint prior to the meeting. This visual demonstration gave PO1 and his supervisor the ability to evaluate their own requirements and to judge the work of the developers.

The meeting was held through an online collaboration tool, WebEx, which allowed the PM1 and the Gera representatives to view the same running instance of the Statistics in real-time.

The product manager M2 in Stockholm attended the Sprint review meetings a two–three times, just to observe and to get an insight on the progress of the project. M2 was concerned about the lack of a long term plan, and felt that the project should have had more focus on future goals.

Daily Work Flow

Each day of development started in Trondheim with a morning meeting at 9 am which lasted for about 10–15 minutes. Only the technical development team would be present at this gathering, not the product owner. The developers would quickly go through what they had been working on since last meeting, and discuss possible issues. If something came up in relation to the backlog or any other aspect of the development that needed urgent feedback

from PO1, the project manager PM1 would contact PO1 by e-mail or phone directly after the meeting. If the issues were of a bigger order, PO1 would consult his supervisor.

PM1 would sometimes ask PO1 to include visual examples by e-mail, in order to clarify issues. These examples were often screen shots from other sites or rough mock ups of the concept or idea PO1 was trying to convey. This communication then got forwarded to the developer having initiated the process. PO1 would rarely communicate with the developers directly. He feared information overflow if he were to work out issues with the individual members on a daily basis.

7.3.2 What Happened During the Development?

I will now go through some key features that was implemented during the first four and a half month of development.

A New Navigation

The first Sprint began on June 14. The development team consisted in Trondheim at this point of five summer interns, four working as developers and one working as interface designer, and three permanent employees of Sportradar. These were filling the roles of one system architect SA1, developers and a project manager PM1.

The new navigation should make it possible to navigate from a continent to a specific league, for example from Europe to the German Bundesliga. It should be done with three clicks on the mouse, hence it had been doubt the *three level navigation* by the Gera department.

The feature description of the three level navigation had been added to Redmine in April. Since then, PO1 had added several documents as attachments to this feature, including some conceptual screen shots and a 30 pages long word document describing the navigation.

The workload of implementing the new navigation was originally estimated to ten working hours in the first Sprint. This was later discarded as the solution did not meet with Gera's requirements. The work on the navigation system went on to end of Sprint 3, when it finally got approved with the perceived need for only minor adjustments.

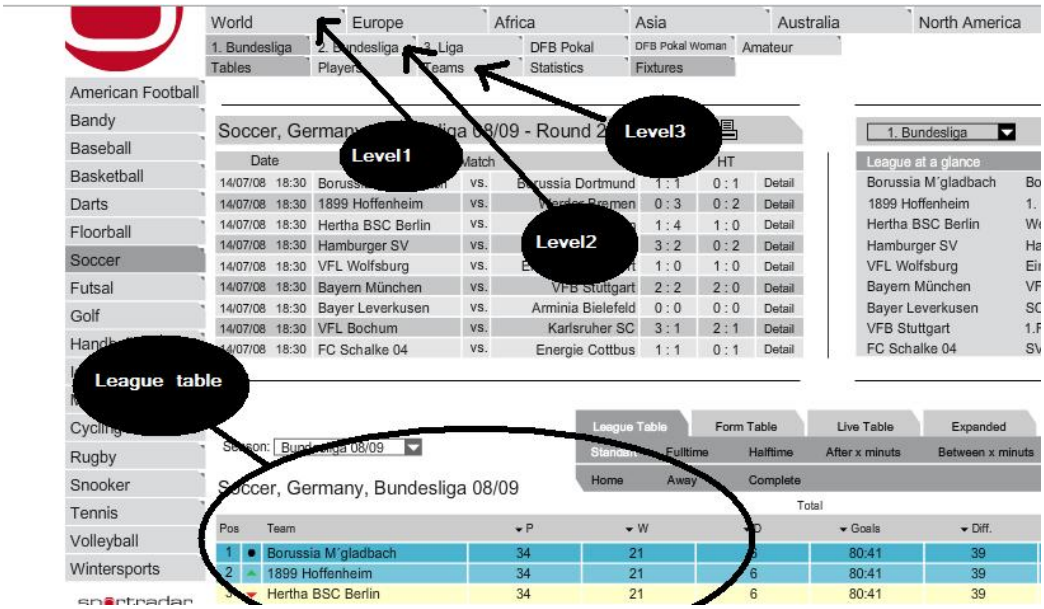

Figure 7.3: Figure showing the three level navigation, and the league table functionality in an early version of the new Statistics

Data Components

The old Statistics was built up by various components, which also should be featured in the new system. One of the additional component Gera brought into the development was the introduction of new fields in the various tables. As the data base were of high complexity, introductions of new fields would cause a major programming effort on the development side. It would also requires domain knowledge to know how customer specific components are built, something the summer developers did not have.

The data components and related features requested by Gera was not perceived as a realistic amount of implementation work by the Trondheim team. An agreement was made to focus on support for two sports only: mainly soccer, and later tennis. These sports should be supported with components and fields that had its equivalent in the old Statistics. This was to give the developers time to create generic code. The systems architect SA1 was the advocate for this decision, and the team supported his argument. His long experiences with the Sportradar databases and the various systems that had been developed over the years made his voice quite prominent in the Sprint planning meetings.

The Search Option

The planning meeting for Sprint 3 was held on July 17. Gera wanted the developers to start working on a *search option* which should make it possible for users to search for players and teams. This was considered as very important by Gera. The product owner PO1 had created a Redmine issue describing the feature together with a 20 pages of additional documentation during Sprint 1.

The search option did not exist in the old Statistics solution, and the request came as a surprise for the developers. They had perceived the agreement made in the first Sprints about keeping to the old Statistics functionality until it was stable, as a safe guard against unfamiliar implementation work. The project manager PM1, who had stressed the value this option had for PO1, wanted to explore the possibility of including the search function. The system architect SA1 was on holiday when this came up. SA1 would normally have the responsibility for estimating any large feature, but this task was instead given to one of summer job developers. After a day of research, this option was put on hold due to the complexities it involved. From Gera's point of view, this was not ideal, but they respected the decision.

Work continued on data components in Sprint 3, and the developers were making headway. As the level of functionality grew steadily, Gera oriented towards the visual qualities of the various graphical components the interface now contained. It was perceived as very important for PO1 that the solution should stand out, and this resulted in effort being put into making various components more pleasing to the eye.

> "The media client really looks at [the data] to see if it is correct and how [the web page] is presented in order to generate more clicks. So, based on this, the media market was more important. Of course we concentrate on the bookmakers as well. [..] The is a challenge of the project, to get a line between both markets, to find a way to satisfy both of them."
> — *Product Owner PO1*

A feature that got demonstrated in the third Sprint review was a graph that displayed the tendency of a team in terms of victories over time. The functionality had in Sprint 3 had textual description in Redmine. The developer that worked on it had no visual reference to go after, and the result was not what Gera had had in mind. PO1 *requested another version with 3D-graphics*, a requirement the developers had troubles grasping. By providing

a visual conceptualization to PM1 by e-mail, PO1 succeeded in conveying his idea to the developer working on the 3D-graphics.

As the design and navigation system got added to the system by Sprint 3, the first version of the new Statistics was up and running—although still rough in the edges—on an internal web server in Trondheim.

The first reactions to the early version was in Trondheim somewhat mixed. The navigation and the user interface design got questioned by both members of the development team and Trondheim employees on other projects. Both D3 and GD1 told their disliking to the project manager, and even though PM1 agreed with the critique, he were still following PO1's requests. The attitude of the team went somewhere along the lines of *the customer knows best* and *if that's what the product owner wants, that's what he's gonna get.* Usability was in the end regarded as the product owner responsibility.

Tennis Support

The planning meeting for Sprint 4 was held on August 3. Prior to this, the navigation system as well several important data components had been approved by Gera. Gera was at this point very satisfied with the progress they saw with the Statistics, and the product owner PO1 was eager to include new and more innovative features. As a result, support for tennis was requested.

The development had so far been focused on getting support for soccer when it came to data components. Further, the developers that worked on these had for the most parts mirrored Statistic services that existed in the old Statistics solution. The old system had thereby provided them with a convenient template for what should be included in the new components.

The old system had a limited degree of tennis support, and Gera's request for a full tennis support presented the development team with a real challenge since they could not use the old implementation as a starting point. The tennis effort began as an exploratory mission of identifying what kind of technical challenges that had to be resolved in order to implement Gera's proposal for a competitive (in relation to the market) tennis service. This proposal was formulated in a 30 pages word document by PO1, and sent to PM1 prior to Sprint 4.

The task of analyzing the document and the Sportradar infrastructures in order to gain an overview of tennis became a time consuming endeavor. The task was estimated to be finished within four hours, but it took the assigned developer two working days.

> "[The document] was quite to the point and specific when came to what we needed. The difficult part was to understand all the tennis terminology. For example 'grand slam' and stuff like that [..] There are so many weird terms, not even talking about the number of obscure tournaments. It took me two days, and I asked around the office to try to find out what we had support for. [..] Also, there were a lot of weird abstractions in the database, like 'players' in the same field as 'teams'."
> — *Developer D2*

The result was a set of tennis features in Redmine that the team would work on in the subsequent Sprints.

Rearranging the Team

As work of tennis support gradually began, the development team got rearranged due to the summer job developers starting their university studies again. Two newly hired employees as well as two more experienced Sportradar employees got assigned to the project, while the project manager and the last developer stayed put. The system architect SA1 planed to gradually withdraw from the project as he was needed in other projects. This created some problems because SA1 had become a technical go-to-guy both for Gera and the Trondheim team. He received more or less daily e-mails from PO1 with questions related to the Statistics, as well as other projects Gera was involved in.

Towards Beta

By Sprint 6, which began on September 1, the new development team had settled in, and they worked primary with getting full tennis support. The project was perceived both in Trondheim and Gera to be on track for beta release in November, and in relation to this the project team perceived the need for fine tuning in terms of functionality. PO1 was therefore scheduled for a visit to Trondheim during the first week of Sprint 8.

The Project Owner Visiting Trondheim

PO1 arrived in Trondheim on September 27 and stayed at the Trondheim offices for a full week. He was involved in other projects as well, but would

primary focus his attention on the development of the new Statistics. He worked with several of the developers directly. For fine tuning of the data components, he would walk through screen shots of the Statistics, highlighting were it needed changes. The developers present on these fine tuning sessions found it to be a very efficient way of learning about PO1's requirements, especially since they had had no prior contact with PO1.

Internal Design Team

November 4 an internal team of graphical designers was assembled, assigned by the upper management to improve the graphic design. As the solution was close to being released, management had sen the need for bringing in some competence on the user interface, to "push things forward" -GD2.

The design team consisted of two full time workers and one part time worker, who started on some rough sketches. One of the full time workers was the graphic designer GD2, who started working for the company in October 2009 for the Stockholm office. GD2 had background as a web designer with one year studying computer science, and was specifically hired to work on the Statistics project. From 2010 he were to work full time as the only graphical designer on the Statistics, as the two other graphical designers then left the project.

> "We are a small design group, [..] competing against big design groups, so we have to do some research , we have to go to every competitor and pick up the best from their sites. Choose and make it better. We didn't do [any official] market research. It's more like a gut feeling. When I have two other to discuss with it makes it a lot easier. [..] We now like to keep [the design] within the the design group, because it gets too complicated if they show it to too many people who really don't have the expertise about, like the sales people, who just want to see something fancy. Sales persons don't know anything about design."
> — *Graphical designer GD2*

The manager in Stockholm had clearly seen the need for bringing in a design group.

> "Style sheets should have been included from the very beginning."
> — *Product Manager M2*

November 20 the design team traveled to Trondheim for a couple of days. This was the first time they met each other. Here they did some brainstorming around the design, and came up with a presentation of their suggestion for the design to the product owner PO1, his supervisor and the project manager PM1, who were pleased with the sketches. However, there were no time to implement a new design for the internal release of the new Statistics., but the management were prepared to do some "cleaning up" in the design.

Internal Release

The beta version of the Statistics got released for internal use on November 3 2009. *This was the first time anyone outside of the development team got to see the product.* So far, neither the sales people or upper management had been involved during the development process.

According to PM1, the upper management got a link to the new solution while the sales team got a demonstration on a big screen. The reaction to the solution was not positive: UM1 and sales were not happy with the user interface design and the navigation, and they did not think that Sportradar would be able to sell the product as it was presented at that point.

7.4 Redesigning The User Interface (January–May 2010)

Rather than rejecting the whole product, the management in Gera and Trondheim decided that a new interface should be made. No more features were to be implemented, and the design should be upgraded. The deadline was postponed yet again, this time to April 15 2010, when a demo was to be presented for a new, big media customer. The deadline for the final implementation was set to June 1, 2010.

Shortly after PO1 had presented the current Statistics to the company November 3 2009, the developers in Trondheim got the message at an information meeting that the design had not been approved. No further explanation were given. The project manager PM1 then told them they had to remake the design.

The students and part time workers on the team was not fully informed about what was happening. In the middle of March 2010 GD1 did not knew when the next deadline was, nor did she knew anything about the reception at the

demonstration in December.

> "There have been so many deadlines: it always get postponed [..] Suddenly they were just doing something new."
> — *Graphic Designer GD1*

January 2010 GD2 started designing the new landing pages[2], making some sketches in photoshop that he shared with the rest of the team through Redmine. The layout was to remain more or less the same, but they were to "take the current solution and put it in a new packet" -GD2. The developers and the project manager were excited by the new design, and saw it as a big improvement. "The new, white design looks much better" -GD1. D3 also described the new design as "much better", although:

> "It's a bit overkill with all the fancy design. But then at least the customers get to see that we are able to make something fancy for them, if that's what they want. And still, we have the opportunity to simplify the design."
> — *Developer D3*

opportunity

From February 15 to 19, GD2 was in Trondheim. This was the first time GD2 got to meet the rest of the team. The purpose of GD2 coming to Trondheim for one week was, according to GD2, to establish a connection with the rest of the team.

> "It was necessary for me to come [to Trondheim] and start to get some connection, to know what [the developers] are capable of doing. Are they able to make new features, or just recreate? So I'm trying to get to know them. I have also been working with the stylist [GD1]. [..] I'm here for them to come to me and ask questions. If they need some basic feedback or have some small questions, they can come directly to me, and visa verca. "
> — *Graphic Designer GD2*

Developer D3 became main responsible for implementing the design together with graphic designer GD2, who continued being a part time worker spring

[2]A landing page is the default page that shows up when a user first enters the web page.

2010. Two of the other developers also got assigned issues regarding the user interface design.

GD2 made some mock-up images, passed them on to the project leader PL1, who presented them to D3 and GD1. GD2 made it clear that they should let him know if they disagreed with the design, thereby giving the developers a great degree of influence on the design. Those times GD1 had showed her disagreement, GD2 had listen and changed the design. D3 also confirmed this, and said that he could disagree with GD2's design, and then it would be up to PM1 to discuss a settlement with GD2.

During the redesign of the user interface, PO1 was working with quality assurance, discovering bugs in the system regarding behavior and content of the site. PO1 continuously registered bugs in Redmine, which were assigned to the developers by the project manager PM1 at the daily Scrum meetings. PO1 did no longer take decisions regarding the look of the web page, like colors and placements of tables, text and pictures. However, PO1 still checked for layout problems, like adjustments of columns, and came with some minor styling requests, but then in the form of task or issues together with the rest of the bugs.

7.4.1 From a Sales Perspective

February 24-25 2010, PO1 showed a demo to the sales team in Gera, where he presented the new design. According to one of the sales directors in Stockholm, this was the first time they saw the new Statistics. At that time, the sales directors had been waiting over half a year to see the results, and ideally, they would have seen a demonstration at a much earlier point.

> "What is flexible? We say to our customers that this solution will be so flexible, but everyone says that. You have to be able to *show* the customer a demonstration of the flexibility."
> — *Project Manager M2*

Without having a visual demonstration to show the customer, the sales had a harder time trying to sell the product. Since the Statistics always got postponed, the sales directors in Stockholm had to keep the customers on hold and tell them it would be ready "next month". The sales directors were constantly waiting for a visual demonstration, and were more concerned about pushing the product out on the marked, rather than getting it 100 percent correct the first time. As SD3 said, the media customer would not

expect all the historical data to be 100 percent correct, and minor bugs would be found as soon as the customer started using the product, which could then be corrected afterward.

After the demonstration of the Statistics, the sales directors had to come up with a documents regarding what they would like to see different.

> "I guess [the management] wanted the sales perspective on what should be fixed. [..] They should have involved sales much earlier, so that we could have gone out to the customers with the solution and ask for feedback. Now we don't have time to do much change, so [the management] want us to sell the current solution without much change."
> — *Sales director SD1*

April 27 to 28 2010 four sales directors came to visit the Trondheim office. They all had traveled a long way for this relatively short meeting, from Sportradar's offices in Germany, France, Sweden and Hong Kong. The agenda for these one-and-a-half days of meetings was first and foremost to present sales with the Sportradar products.

The meeting started with a short introduction to Redmine and Medialogin, focusing on how to register issues and requests from the customer. Redmine impressed the sales directors by offering a set of formal priorities of tasks and issues. It also allowed them to follow the evolution of an issue, seeing how many percent of it was finished.

Before all the Sportradar products got presented, the sales directors were given a short introduction to XML, without going too deep into the technical details. The directors had a lot of questions about XML feeds and multicast. The different solutions of delivering the feeds were explained by one of the developers in Trondheim, like the difference between offering a hosted solution compared to XML feeds from the Sportradar servers.

The questions raised by the sales directors at this sales meeting *reflected the most frequently asked questions by the customers*. The reason why the sales directors wanted to know more about XML was that this was something the customers often asked about, and sales were not technical people. It was made a joke about the sales director being confused, but they wanted to have at least *some* basic knowledge about XML, so that they would be less confused than the customers.

Another selling point was the extended information about the players. After the project manager PM1 had showed a demonstration of the old Statistics

and then the new Statistics, clicking through the soccer information, one of the sales directors pointed out:

> ”Compared with the old Statistics solutions, you can now click on the player, and get all relevant information about each player [like goals scored, height, weight, what clubs they belong to etc]. This is a unique selling point.“
> — *Sales director SD1*

The sales directors seemed to have a very clear understanding of what the customers required.

After the introduction to XML, the developer holding this presentation asked the sales directors ”What are the most common customer questions¿‘. The sales directors had many examples and ideas about what the customers were concerned about, like security and password protection and the size of the XML feed. Historical data were also requested. One important point was that the sales people had experienced a need for involving their customers in the technical aspects of the system integration. As it was now, lack of knowledge about the technical details made the customers concerned about the updates of the feed, what exactly the feed contains, and so on. Since many of the media customers do not know what XML is, the sales directors suggested that the customers should be a part of the setup of the XML feed. The point of involving the customer would have been to make the support easier once the system had been properly implemented. Another suggestion was to provide an internal document that project manager PM1 had made about the Statistics to the customers. The sales directors were impressed by the good sales argument contained in this 8 pages long document.

In Swedish Sportnews there were only one man, C1, responsible for deciding how to set up the Statistics at their web page. There were seven technical developers in his department who would take care of the actual integration, but C1 was the only person the sales directors in Stockholm could ask about the requirements regarding the client integration. Previously, having only one contact person had not been a problem, as the customer contact person from 2007 to summer 2009 had been someone with a technical background, who knew exactly how Sportnews wanted to present the sport statistics. The new contact person, however, were a former journalist with no technical background. The sales directors in Stockholm sometimes found it difficult to collaborate with this customer contact, as C1 was more insecure and vague about the requirements than his predecessor. C1 kept changing his mind, and several of his requirements got put "on hold", as C1 were not able to

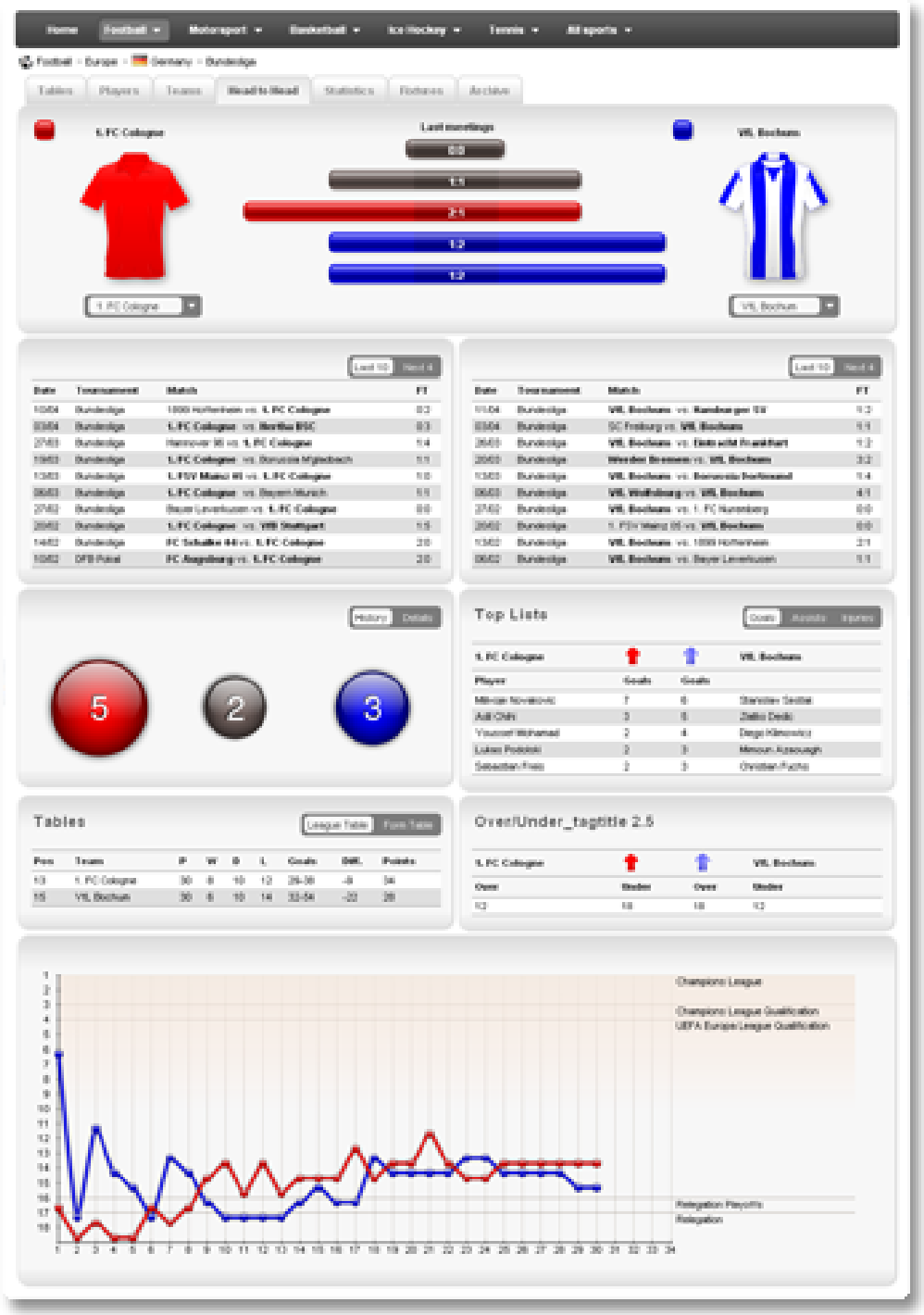

Figure 7.4: Example of functionality form the new Statistics solution.

elaborate on what he meant, or if sales found it difficult to handle. This kind of screening of the requirements was necessary. The sales directors had long experience handling customer requirements, and they knew what was feasible and. The sales directors were able to question the requirements, and on several points, C1 would change his mind.

At the end of my research, by May 2010, Sportradar had delivered the new Statistics to several of their existing customers–to media clients as well as bookmakers.

Part III

Analysis and Discussion

Chapter 8

Analysis

My findings suggest there is a difference between how customer involvement is portrayed in the literature compared to how Sportradar is practicing it. This gap is not unusual, as *agile practices must to be tailored for each specific project* to meet the company's requirements (Abrahamsson et al., 2003; Fitzgerald et al., 2006b; Pikkarainen et al., 2008; Nerur et al., 2005).

Finding the best customer involvement policy for a project is not a trivial task. Companies must find their own answer to *if*, *how* and to what *extend* customer involvement should be carried out. Todays literature does not provide the practitiorares with suggestion for how to adress these challenging questions. Rather, agile literature does not focus on this dilemma and has a simpliefied view: the more customer and user involvement the better (Miller, 2005; DeBrabander and Edstrom, 1977) and the more customer representatives the better (Conboy et al., 2009). The downsides related to the involvement are seldom regarded, while the benefits of customer involvement are well known (see for example Kujala (2003)), and whenever new challenges are brought to light, they are immediately dealt with in the literature. Reducing the degree of involvement, or even terminating it all together, has not yet seemed to be a choice.

Agile methodologies argue for one way of engaging the customer, a way that is not feasible for all software development projects. Factors like time pressure, budget, effort and business bureaucracy related to involving the customer is not taken into consideration. These are the challenges I wish to adress, to be able to provide suggestions for practitionares of customer involvement operating in an organization with contraints.

This chapter is divided into three parts:

- **8.1 Agile Development at Sportradar** In this section I elaborate on how agile principles, with focus on customer involvement, are practiced at the Statistics project compared to how it is described in the literature. I will look at the extend the customer is involved at the Statistics project.
- **8.2 Constraints of Off-Site Customer Involvement** Agile literature talk about the on-site customer representative. However, when this is not feasible, a number of challenges will arise. This section illuminates the challenges that Sportradar faced with their effort in off-site customer involvement.
- **8.3 Framework for Managing Customer Involvement** In this section I present a framework for how to deal with the challenges related to customer involvement. I provide a set of recommendations an organization should be take into consideration when practing customer involvement in agile software development.

8.1 Agile Development at Sportradar

Pikkarainen et al. (2008) states that Scrum, ideally, should be implemented at all organizational levels, to allow the team to rapidly respond to the changes in their project's ecosystem . Boehm and Turner (2004), on the other hand, argue for a combination of agility and plan-driven discipline. In their tutorial study, Boehm and Turner (2004) discovers that if the plan-driven management and the agile development team are aware of each others practices, they can learn to coexist.

In the management of Sportradar and other departments than Trondheim, agile was not applied.

> "We don't use Scrum in Gera and there is no need for us to use it. We work differently. But for them here [in Trondheim] I think it's really good."
> — *Product Owner PO1*

When Sportradar first started using Scrum in Trondheim spring 2007, the methodology was not explained to the stakeholders and Sportradar employees outside of the Trondheim office. They did not know how the developers were using Scrum, or even what Scrum was. This *bottom-up* introduction of agile

culture is common in the software industry (Boehm and Turner, 2003b), where the developers are enthusiastic about trying out Scrum, while the management often have an "it can't hurt"-attitude.

Project management is a support function for efficient software development. Still, most agile methods does not offer adequate management support (Cohn and Ford, 2003), which leads to the risk that the customer and the developers lose focus on what the overall purpose of the system is (Valkenhoef et al., 2010). The management felt that they did not monitor the Statistics project as often as they should have done in order to gain sufficient overview on the progress.

> "It's [the management in Trondheim's] role to follow up on the projects, to stay oriented about what's going on. It's not our job to say no, but to come with arguments and point out the consequences of each decision. [..] Ideally, Trondheim should have foreseen some of the problems, but should we have that role? Here, we don't have any experience with sales, with economy, et cetera."
> — *Manger M1*

No one outside of the team of developers, project manager and product owner were included in the development process. Stakeholders were not granted permission to the ongoing process, and vica versa; The system developers in Trondheim did not know what was going on in other departments at Sportradar. "The developers have no clue on what is going on in the management level" — D2.

This lack of involvement from the managers affected the development process in several ways: It affected the Scrum meetings, it led to document based knowledge transfer, it prevented stakeholders from being involved, especially in the requirement specification. It also led to a focus on certain individuals, who were responsible, instead of the collaborative ownership of the project, that agile suggests.

This section provides a discussion about how Scrum principles, are put into practice at the Statistics project, especially focusing on customer involvement. Based on this discussion

More specifically, there are especially four points that should be of interest:

- Customer Representative
- Communication

- Iteration Planning and Review Meetings
- Eliciting Requirements

8.1.1 Customer Representative

XP specifically states that the customer representative should be on-site and constantly available (Cockburn, 2007), as close interaction with the customer and frequent customer feedback are critical success factors (Lindvall et al., 2002).

Operating with customers spread all across Europe and with the development office being in Trondheim, *having an on-site customer representative was not a feasible option for the Statistics project.* Moving the development office closer to the customers were never an option because of Sportradar's desire to utilize the best available development staff, which was believed to be found in Trondheim. But finding customer representatives in Norway with the necessary domain knowledge was more difficult because they had no customers there. However, in Germany they had employees with in-depth knowledge in the customer domain and they already had experience with off-site customer representatives before so they had no reason to believe this would not work.

Finsterwalder (2001) argue that projects can work well without an on-site, full time customer, *as long as the developers get an insight of the domain from the customer, and later can contact the customer if they should have any questions.* PO1 provided the developers with knowledge about sport, though mostly communicated through documents. PO1 was also available for questions, even though most of these questions had to go by PM1 and SA1.

One of the challenges with agile development is to deal with *unclear definition of roles*, as oppose to an a plan-driven culture, where roles and each person's tasks are well defined (Boehm and Turner, 2003b). Scrum also define a set of roles, but these are more flexible, and each person is expected to do whatever work necessary for the project to succeed.

The customer representative at the Statistics had the Scrum role product owner. The product owner at the Statistics had a rather unclear role and many responsibilities. Table 8.1 highlights the diverse nature of the customer representatives role. Some of the roles were constant throughout the project, like *team member* and *boundary spanner*. Other roles lasted only through certain phases of the project, like *product developer* (during planning phase)

Table 8.1: The roles and responsibilities of the product owner.

Role and Responsibility	Related challenges
Product developer	Creating the product description along with detailed specifications.
Representative for user needs	Responsible for ensuring usability
Boundary spanner	Facilitating collaboration across physical borders and domains
Team member	Responsible for being a active part in the development cycles by providing feedback and suggestions
Domain expert	Providing feedback and doing acceptance testing
Quality assurance	Finding bugs in the system.

and *quality assurance* (during spring 2010).

This spectrum of responsibilities is not unique in agile development, but nevertheless, it is not a desirable situation for the customer to be in. It has been shown that out of all the team members, the customer has the most stressful role and most pressure (Martin et al., 2004). One reason for this is that multiple roles and responsibilities can be wearing (Martin, 2009), and because the multiples roles a boundary spanner must have can be conflicting, which again lead to stress and burnout (Levina and Vaast, 2005).

In addition to the demanding nature of the customer role, there is also the challenge of integrating the needs and requests from a diverse user population into the development cycle *having only one single customer representative*. This put a lot of pressure on one person, in contradiction with the agile culture which emphasize a greater lever of shared responsibility (Hanssen and Fægri, 2006; Kane et al., 2006).

There are different degrees of customer involvement associated with the responsibilities of the customer. The more responsibilities the customer has, the more they are able to affect the outcome of the project. One can argue that the more responsibility the customer has, the more their needs are likely to being integrated into the process, and thus the higher degree of customer involvement there is. However, this is not a rule without modification.

Product owner PO1 at the Statistics project had a large number of important roles, but because of who he represented, namely an in-house customer representative, the overall degree of customer involvement is not very high.

An example from the case study of a significant shift in roles can be found when PO1's supervisor went from being the project leader to gradually phasing himself out and taking a more anonymous role in the project. At the same time, PO1 gained a more leading role and became one of the main people in charge of the project. This unannounced change of roles led to a great deal of confusion and disagreement about who was responsible for the project, as illustrated in Figure 8.1. Asking different people about who they though of as the project leader of the Statistics, I got five different answers. From the management in Trondheim's point of view PO1 was seen as the project leader, and was hence held responsible for the progress of the project.

The Statistics team in Trondheim, on the other hand, did not see PO1 as a project leader. "I think of him [PO1] as the customer. He tells us what he wants, and we make it" — D3. "I have never heard [PO1] being addressed as the project leader" — PM1.

Needless to say, there was no clear assignment of the person in charge of the Statistics project, and "no one wanted to step in and claim responsibility for the project" — M2.

8.1.2 Communication

Agile methods seeks to *minimize the number of intermediates in order to support knowledge transfer* (Korkala et al., 2006), and the number of intermediaries should thus be kept to a minimum, preferably zero. At the Statistics project, on the other hand, there were relatively many intermediaries between the customer and the developers. Often, there would be three links between C1 and the developers: requests from the customers went through the sales agents and one or two managers before it is assigned as a task to a developer (see Figure 8.2), a path which resembles the communication chain found in a Tayloristic approaches.

The lowest number of intermediaries was found in the latest phase of the development, when the customer C1 was in direct contact with project manager PM1 and system architect SA1, in accordance with agile. However, having all Sportradar customers in direct contact with the developers would simply be infeasible; There are several hundred customers; It would cause interruption upon the development process; It could lead to a great diversity in requirements and requests; Also, the media clients—who are mostly non-

Table 8.2: How the product owner satisfy the agile characteristics of a customer representative.

The agile customer	**The product owner at the Statistics project**
Authorized (Boehm and Turner, 2003a)	In accordance with agile. PO1 was given the authority to make decisions, usually on behalf of his supervisor. At the same time, he sought approval by the project manager.
In possession of domain knowledge (Cockburn and Highsmith, 2001)	In accordance with agile.
Representative for the customers (Cockburn and Highsmith, 2001)	No. Although PO1 had a background as a former customer, his requirements did not resemble the once that the customer C1 (or the management) had. However, this is not so strange, as the market segment is diverse, and can hardly be represented through the eyes of one person.
Committed and actively participating in the development process (Nerur et al., 2005). Do the necessary homework Boehm and Turner (2003b).	In accordance with agile.
On-site (Cockburn, 2007)	Off-site.
Constantly available (Cockburn, 2007)	Partly available. PO1 had other commitments in Sportradar in addition to the Statistics project. However, PO1 had frequently—almost daily—contact with PM1.

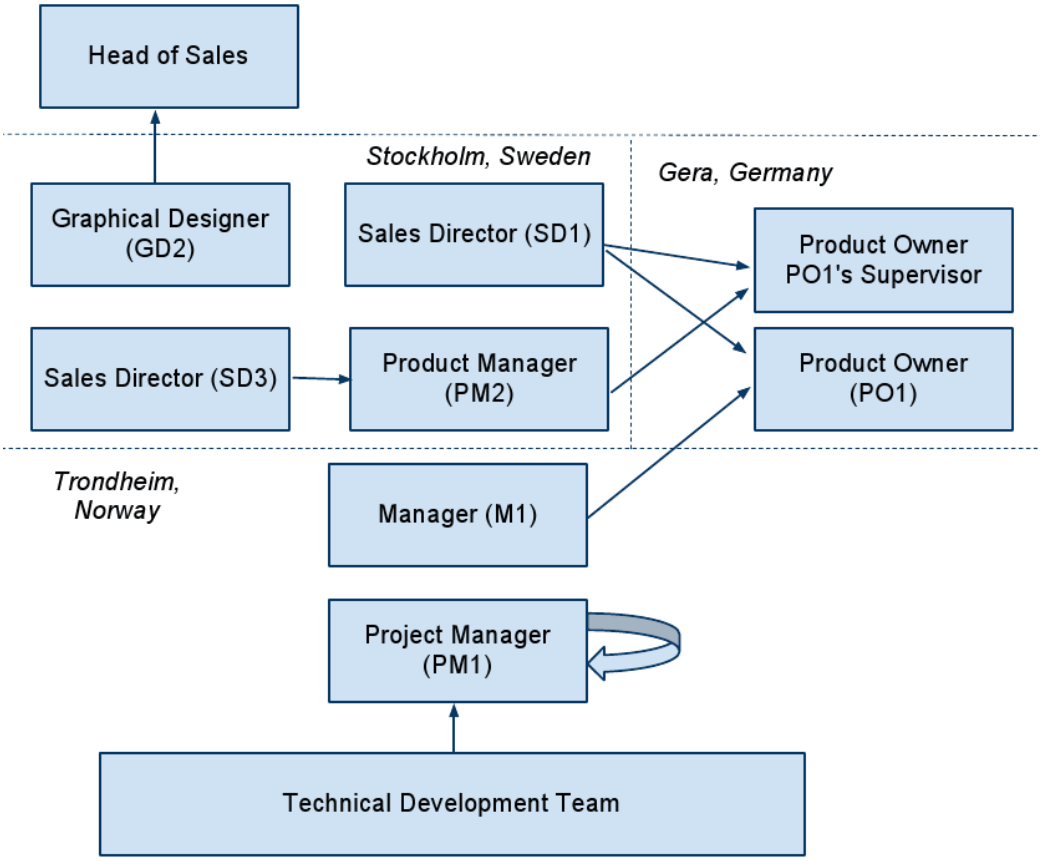

Figure 8.1: Illustration of the confusion and disagreements about who was in charge of the Statistics project. The arrows are pointing from one person to who that person thought of as the project leader.

Figure 8.2: *The intermediaries between the customer and the developers*, illustrating the worst case scenario when it comes to number of communication links carrying the message of a *customer request*.

technical—might not feel comfortable expressing themselves to a technician, like C1 who preferred to communicate with one sales agent mostly.

8.1.3 Iteration Planning and Review Meetings

Besides the creation of the backlog, other arenas for building a common ground between stakeholder are the planning and review meetings, where both customers, management and developers are supposed to participate and contribute with insights (Schwaber et al., 1995).

This decision of not involving the developers in the creating of the backlog and at the planning meetings can be explained by the high turnover of developers. Firstly, the summer interns were set to work on the project while it was clear from the beginning that most of them would leave the project after only two months. Secondly, who was working on the project seemed to be adjusted while the project went on. Two additional persons joined the project after the summer, and Sportradar also planned to employ two new developers by September 1 2009. Three of the summer interns decided to follow the project as part time workers throughout the fall (only one of these continued into 2010), while the five remaining interns left the project. This leaves only three developer plus the product owner left as the four remaining team members who have participated from the beginning to the end of the project.

Lack of stakeholder involvement can also be explained by the *fear of exposing the concept*. According to employees at the Stockholm office, product owner PO1 did not want to involve stakeholders other than the development team in Trondheim, in fear of exposing an unfinished product. Reluctance towards involving stakeholders can also come from a feeling that no input and no further feedback is needed, since all the required domain knowledge is already present in the team. "I know what [the customers'] major interests are." - PO1.

Not making the backlog creation an arena for discussion and knowledge exchange can point to incidents like the rejection of the *search option*. This functionality was part of the initial backlog, but came in Sprint 4 as lightning from clear sky for most of the development team. This could have been avoided if the developers had been part of specifying the initial backlog together with the customer. If we look back to the developer working on the tennis requirements in Sprint 5, it is interesting that he did not contact PO1, although he was struggling to comprehend aspects of sport related issues. As the developer never had interacted with PO1 before or participated in

discussions were he was involved, this is perhaps not all that strange.

Another consequence of not making the demo meetings open for discussion was the sales were not given the opportunity to give feedback. No one told sales in Stockholm about the new features of the Statistics and its selling points. Sales simply got a link to the hosted solution, and had to start working out the selling points themselves.

> "We did not have a product owner who sold us [sales] the Statistics. We had to find the selling points ourselves, by clicking around the page."
> — *Sales director SD1*

8.1.4 Eliciting Requirements

Transfer of knowledge and realistic expectation of what is possible to achieve between customers and developers is a vital part of agile development. Agile methods seeks to minimize the documentation found in traditional methods to support this knowledge transfer by introducing activities that enhance direct collaboration with the customer. In Scrum, the creation of the initial product backlog can be said to be such an activity, as it in an early stage is intended to bring the development team together with stakeholders in order gain a common conceptual understanding of what should be built (Korkala et al., 2006; Paetsch et al., 2003).

Before the Statistics project went on to its implementation phase, the product owner PO1 created the initial backlog. PO1 did this mostly on his own, contradictory with Scrum, which specifically suggest that the backlog must involve all participant.

Why Sportradar chose not to involve stakeholders other than the product owner and SA1 in creating the initial backlog have several reasons. First of all, the team was not assembled yet, and the Trondheim office has too many developers that it would have been feasible to include everyone. Involving a few random developers would have been risky, since they would possible be needed on other projects after the specification had been made, thus bringing the knowledge with them and out of the Statistics team.

More planning requires more involvement from all the stakeholders in the crucial planning phase of the project. However, the stakeholders on the Statistics were not involved in the creation of the backlog. However, the backlog is not suppose to work as a final plan, but should rather be re-prioritized after each

demo. This gives the opportunity for stakeholder feedback after each iterations. However, since even the internal releases were few and happened in the later stages of the process, the stakeholders were not given the chance to participate.

One of the focal values from the agile manifesto is "working software over comprehensive documentation" (Fowler and Highsmith, 2001). Focus should be on communicating effectively and documentation should be the last option (Lindvall et al., 2002). In Scrum, writing full requirements up front is discouraged, as the user is not expected to have a concept of what is possible and the developers do not at this point know fully what can be built (Beedle et al., 1999). In the Statistics project, however, product owner PO1 spent a great amount of time specifying the initial requirements before the implementation work started. PO1 started working on the initial requirements in October 2009, and five-six months later he traveled to Trondheim and created the initial backlog in Redmine.

The first principle of the agile manifesto states that "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software" (Fowler and Highsmith, 2001). This might be a reasonable goal for smaller projects, but for larger systems, over-focus on early results can lead to major rework when the architecture does not scale up. In such cases, a good deal of planning will be necessary (Chau and Maurer, 2004).

It is not unusual in agile practices for the customer to be responsible of defining requirements prior to development. Sportradar is thus not unique in this situation, but they do differ from agile practice by not having the developers be a part of making the written concepts. The case study presented by Hanssen and Fægri (2006) (see page 38) brought out the importance of agreement and mutual understanding of the implications of agile development between the various participants. The important role documentation played, intended from PO1, in the early phases of the development can come from a lack of such agreement and effort to build an understanding of the process. At the same time, relying on documentation alone can be a hindrance for mutual understanding, as detailed specifications serves little purpose in itself if the development team are working in agile iterative fashion where the developers expect to gain knowledge by direct communication.

The documentation containing the initial requirements had a *high level of details* of the specifications. One example of this was the deep linking concept, which was just one of many requirements from the comprehensive document of 200 pages. While the main requirement from sales were that the product should be "fancy", the initial requirements were of a much lower level.

Much of the documentation were never read by the developers - "no one told me about the documentation in Redmine" -D2. Agile specifically ask the practitioners to be aware of warning signs like the production of "useless documentation" (Lindvall et al., 2002).

8.1.5 Summary of Agile Versus Sportradar

Given team size, the need for customer involvement and the nature of the Statistics project, Scrum is a suitable method. But like with most projects, Sportradar made some adjustements and did not apply all recommendations and principles stated by agile literature.

Table 8.3 summarize the differences between Sportradar and the literature in a table. Regardless of whether or not Sportradar's appliances are causes of conscious decisions or merely coincidences, there is an explanation behind each practice. These reasons can be found by looking at the *trade-offs* Sportradar's had to make when they made their decisions.

8.2 Constraints of Off-Site Customer Involvement

Agile literature argue for an on-site and constantly available customer representative who provide the developers with ongoing expertise (Cockburn, 2007). Having an on-site customer representative facilitate communication and makes it easier to collaborate and integrating customer needs into the requirements. Whenever the developers have a domain specific question, they can simply just turn their chair and ask the customer representative directly.

However, as we have seen in section 8.1, having a fully available on-site customer is not always achievable. And when the ground basis of agile activities are not present, many of the agile principles becomes harder to fulfill. Dealing with an off-site customer representative is one of these circumstances that complicates the agile development, and thus requires more caution when organizing the team and adapting to agile.

Deciding on what activities supporting customer involvement to apply in an agile team is a *trade-off between effort and cost and the value gained in terms of increased customer satisfaction.* Whenever a trade-off is being made, the organization might chose to give up some of the benefits of having a high degree of customer involvement, but at the same time they will gain in

Table 8.3: Agile software development at the Statistics project

Practice	**Agile**	**Sportradar**
Documentation	Focus on running code over excessive documentation.	The initial concept was described in a 190 pages document; Description of feature were communicated through documents.
Pre-game phase	Planning of requirements are done in close collaboration with the customer. Requirements are gathered from all stakeholders.	Planning was done by PO1 and his supervisor alone.
Backlog	Should initially contain high level goals. All stakeholders contribute to the backlog. Features are estimated and prioritized.	High level of details. Only PO1, his supervisor and two of the developers contributed. No systematic prioritization.
Sprint review meetings	A demo is showed to stakeholders at the end of each Sprint.	PO1, his supervisor and PM1, sometimes SA1, were the only people present.
Sprint retrospect meetings	Held at the end of each Sprint.	Not applied at Sportradar.
Customer availability	The customer representative should be fully dedicated and constantly available.	Product owner PO1 was not one hundred percent dedicated to the Statistics project.
Decision making	Collaborative decision making, involving the stakeholders (Nerur et al., 2005).	Low degree of collaboration.
Acceptance testing and focus groups	Often used in agile methodologies to systematically gather customer input.	No systematically gathering of customer input.

another aspect. Finding the right balance requires the decisions to be made with full comprehension of both the negative and positive aspects.

This section explains the dilemmas the Statistics project encountered. It discuss the consequences and challenges Sportradar faced in dealing with an offsite customer representative, numerous intermediaries and lack of input and feedback from the end customer—all consequences of the choices Sportradar made combined with the given context the organization was operating in.

The trade-offs I wish to elaborate on evolve around the following topics:

- Communicating Trough Intermediaries
- Involving the End Customer
- Obtaining a Shared Understanding
- Stakeholder Involvement

8.2.1 Communicating Trough Intermediaries

Communication and knowledge transfer becomes a challenge when there are numerous links between the end customer and the developers. As we have seen from the previous section, in the Statistics project there could be as many as three links between the customer and the developer. Having this many links can cause a problem for communication, as seen in Chau and Maurer (2004), who explains how longer communication chains leads to a drastically higher degree of information loss. Since agile methodologies disregard documentation and rather rely knowledge transfer through direct face-to-face communication, agile requires the organization to facilitate direct communication between team members and stakeholders.

At the Statistics project, information loss is a possible efect of having up to three intermediaries between the customer and the developer. For example, the developers frustrations and objections to the interface design did not reach PO1. Another example is the end customer who came with requests to sales, which sometimes were disregarded by the sales directors (see figure 8.3).

There are not just downsides to having intermediaries. At Sportradar, *intermediaries were used to filter information.* The project manger PM1 had the role of an intermediary between the developers and the product owner, filtering information each direction. Communicating through PM1 was reasonable, since having direct links between all developers and PO1 would cause an

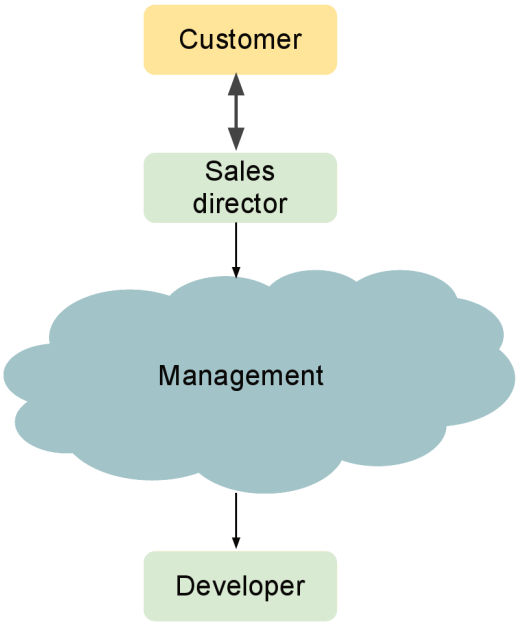

Figure 8.3: The communication flow between the customer and the developers were limited. SD1 had close contact with C1, but the developers mostly related to the management.

overload of requests intended for PO1. With PM1 managing all the requests, *PM1 worked as a boundary spanner*, and translated the developers' requests into understandable language for PO1, who was a non-technician. Because of this, PO1 was comfortable with the way the communication worked.

8.2.2 Involving the End Customer

Besides the convenience of developing something not completely new and unknown, another explanation for why not more stakeholders were not involved in the Statistics was the dilemma of deciding on *who* to involve. Find a representable customer voice is not an easy task. Sportradar operate in a diverse market with several hundred customers and even millions of end users, and there are many opinions on how the Statistics product should look like and what functionality it should provide.

Deciding on what customers to involve would have caused a great deal of effort, and considering the risk of receiving skepticism and reluctance from the customers, it was more convenient to not "bother" the customers with this request.

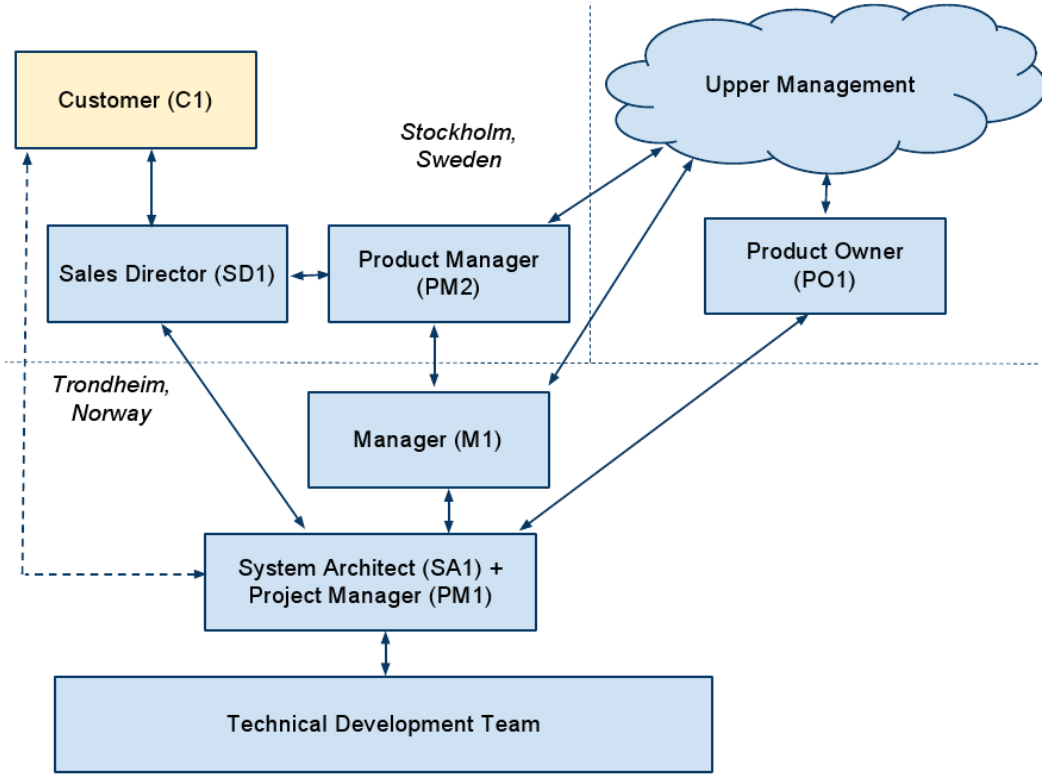

Figure 8.4: The communication flow between the stakeholders at the Statistics, demonstrating a restricted link between customer and developers.

During my case study I discovered reluctance from the customer C1 towards being more involved in the development process. The reason for this, C1 explained, was that he did not want to allocate resources for one of his colleagues to be available for Sportradar. Secondly, C1 did not think that the benefits of being part of the development process would out weight the loss of competitive advantage, as Sportnews's competitors also would benefit from the improvement of the product.

> "There is a line. There should not be too much customer involvement, as this takes too much time, energy and resources."
> — *Customer C1*

C1 was satisfied with the way the Statistics project was running, and had not wanted to be involved at any earlier stage of the development. C1 preferred to be involved in the later stages, where he could do minor adjustments and tailor the product to fit his company's needs. That way, Sportnews's competitors would not copy their solution.

From a customer's point of view, *allocation of resources* and *loss of competitive advantage* are two of the biggest hindrances for getting involved in the development. For Sportradar to involve the end customer, they would have had to convince the customer that the value of being engaged in the development would out weight the effort and risk involved. The challenge of how to account for the customer's motivation for willingly sharing their views and collaborating with the organization is seldom mentioned in the literature (Lundkvist and Yakhlef, 2004).

Before the company can convince the customer to get involved, they need to be convinced themselves. Lack of knowledge about customer involvement, reluctance for change and the effort of employing a new practice are all hindrances for getting started with customer involvement in an agile context. Together with skepticism from the customer, these are some of the reasons why customers are still being left out of the development in many projects, in spite of the common agreement in agile literature about the necessity for customer feedback into the development cycle.

8.2.3 Obtaining a Shared Understanding

Having an off-site customer representative pose certain challenges, like facilitating communication and the costs of time and money spent on traveling. It is also in conflict with agile method's focus on face-to-face conversation,

which is claimed to be "the most efficient and effective method of conveying information with and within a development" (Fowler and Highsmith, 2001), and to transfer ideas faster, thus saving money and time (Highsmith and Cockburn, 2001).

Given that the customer representative has to be off-site, some challenges emerge when it comes to collaboration and communication. Due to lack of physical meetings, the requirement from PO1 was mostly communicated through con calls with the project manager, trough e-mail with SA1 and through documents and issues registered in Redmine.

There were several incidents at the course of my study where communication led to misunderstandings. The first example in the case study is related to the deep linking-requirement. Developers in Trondheim thought that deep linking meant integrating a subset of data components into the clients web page, while the sales director understood deep linking as an URL with variables. PO1 used the term "directly link to the page deep within our data world", which is not a very concrete and specific definition of the concept. A second example of confusion can be seen in section 7.3.2, where PO1 requested 3D-graphics, and the developers did not understand what he meant. The developers not understanding the domain language come across as a problem when the developers were to implement support for tennis into the Statistics solution. PO1 had written the requirements by using tennis specific terminology, which the developers were not familiar with. The *ambiguous language* in these examples caused misunderstandings and extra effort and time during implementation. The ambiguous language causing these situations can be related to the non-technical product owner.

> "I get all my Redmine mail. [..] I try [to use Redmine to follow the progress] as much as I can. It's not always possible because you just don't understand what they write there. [..] But I'm not a technician, so it wouldn't have made any sense to me to be included more [in the technical process]."
> — *Product owner PO1*

As supported by this case study, having an end customer *with technical skills* has proved valuable. One example is the former customer contact from Swedish Sportnews, who was resumed by C1 summer 2009. The previous web administrator's deep technical skills was a quality the sales directors in Stockholm appreciated, as made communication easier. C1 on the other hand, had background as a journalist with limited technical skills, making it

more challenging for sales to cooperate with the customer. Technical knowledge was a property found at most of the bookmaker clients, and they were of this reason perceived as easier to communicate with than the media clients, according to the sales directors.

> "The media clients are very different from the bookmaker clients. Media clients often do not have a technical background. [..] We need to make use of more visual tools to find out what the customer wants."
> — *Manager M2*

The fact that the product owner and the developers sometimes misunderstood each other led to extra work and frustration. Incidents like the *deep linking* concept would probably not have happened if technicians, sales and domain experts had had the same terminology. This example illustrates the advantage of having the domain expert and the technicians to speak with the same terminology.

> "[PO1] should also have a technical background, not just be a domain expert. [..] That way you can avoid having the [product owner] write something about the domain and expect the technicians to understand it."
> — *Manager M1*

However, agile literature does not require the customer representative to speak the same technical language as the coders. Instead, it is the developers' responsibility to *translate agile and software issues into customer language* (Boehm and Turner, 2005). To translate the technical language, the developers are requires learn to understand a sufficient amount of the customer's domain language as well, enabling true two-way communication.

8.2.4 Stakeholder Involvement

Stakeholder involvement is an essential practice in agile methodologies (Nerur and Balijepally, 2007), and the development should—at least—include representatives from the customer, developers and management (Eberlein and do Prado Leite, 2002).

At the Statistics project, the customer, sales and management were not engaged in the development process until after the implementation phase. This

group of stakeholders first got involved after the internal release, when the upgrading of the user interface and tailoring of the product was done spring of 2010.

Why Sportradar did not involve more stakeholders in the earlier phase of the project can be explained by the company culture and the context of the development. Sportradar had developed products for customers since 2001, and there were not anything revolutionary with upgrading the Statistics. It is thus understandable that the management felt they were in control of the process when making another product in a known domain, and did not look for more stakeholder input to the development. It was also natural to assign PO1 as the product owner, with regards to his background as a former sports journalist and knowledge and experience with the old Statistics. The project were thus seemingly in good hands.

Also, the management in Trondheim showed an attitude of not wanting to interfere the development since they saw PO1 as the project leader and trusted him to be fully in control over the project. This might not have been a conscious decision at the moment, but management realized after the internal release that they should have followed more up on the project.

The biggest challenge with not involving other stakeholders is *getting the requirements right*. Getting the requirements right from the beginning requires that the people making the requirements know the customer domain. This was partly true for the Statistics project; Or at least, that was how the management and product owner felt it, which can explain why stakeholders were not more involved.

Autonomy and Ownership

During the first phase of the project, the implementation in 2009, the developers were reluctant to speak up to the product owner whenever they disagreed with the implementation and interface decisions being made. PO1, however, seemingly intended keep the decisions open for discussion.

> "To tell you the truth about the feedback, there wasn't too many questions. It must be that the building block-definition was okay and they [the developers] have an understanding of what we are doing. Plus we are open for ideas from them [the developers] as well. I mean, it's not like we have a concept and we have to stick to it. We are open for their ideas and their solutions, because they have experience in this. This concept is the *guideline*, but

it's not like it has to be copied one hundred per cent."
— *Product owner PO1*

Still, the developers did more or less stick to the initial concept, and did not object to PO1 even though they expressed their frustration and skepticism towards the design inside their development group. Sales, who saw the concept before the development started, did neither try to impart their views, claiming "there was no room for feedback on the concept" -SD1.

This lack of feedback to the management and to the product owner is likely one of the results of not having an official direct communication channel between developers/sales and the product owner. The negative feedback from the developers stopped at the project management level, as there were no mechanisms to fetch up this kind of information and pass it on to a higher level of the organization.

> "If I wanted feedback I barley talked to the developers, because they don't know me. [..] It doesn't make sense [for me to have direct contact with the developers], because this would end up in constant communication. [..] I wouldn't have a problem if someone else [other than PM1 and SA1] had contacted me, but I would have had a problem if five guys at once had contacted me. Because they all thin, before reading something, 'contact him and ask him'."
> — *Product owner PO1*

Given that PO1 had a very busy schedule and many demanding roles, it is understandable that he did not wish to respond to more input that he already did. PO1's concern that he would be overloaded with feedback from the developers if they all were to contact him gives an explanation to why the communication mostly went through the project manager.

Communication through boundary spanners, like PM1 and SA1, can be an efficient mean for filtering information. On the negative side, intermediaries undoubtedly block direct communication and collaboration between developers and the customer representative, hindering the efficient knowledge sharing agile development requires. This is supported by the Sportradar case study, where poor communication led to several challenges during the implementation phase.

Issues caused by lack of collaboration can potentially provide a breeding ground for friction and frustration, resulting in the *loss of team morale*

(Boehm and Turner, 2003b). Thus, the developers could come across as unmotivated and careless by not taking more responsibility of the Statistics product. However, if the management does not support vertical communication in the organization, the team will feel less empowered since their voice are not being heard.

In agile development, the team should be treated as a partner with both customer and the management (Conboy et al., 2009).

8.3 Framework for Managing Customer Involvement

In this section I propose a set of initiatives an organization can take in order to improve the efficiency of their practices of customer involvement. My aim is to provide a set of recommendations to practitioners for how to make conscious and beneficial estimates of the trade-offs with different preparations for customer involvement. I base my recommendations on the lessons learned by the Statistics project, as well as the literature I have researched.

These first questions the organization should ask itself is:

> *How much domain knowledge is needed to build a system that satisfy the system requirements, and how much is already present in the development team and in the organization?*

Answering this should give an indication on how much customer input and feedback is needed, which further leads the organization to answer what customer representative to choose—if any.

Thus, the second question to answer is:

> *What alternatives for customer representatives are there, and what alternative is most suitable for the specific project?*

Based on the challenges I have observed at Sportradar together with the literature presented in Part I, I have created a list of topics that an organization need to reflect on in order to obtain successful customer involvement. I will now present these topics together with discussions of possible measures that an organization may implement related to each topic as a framework for customer involvement.

- **Whom to Represent the Customer**: There is no one-size-fits-all solution to whom to use as customer representative(s). Choosing whom to use as customer representative is therefore required to be done on a per-project basis depending on the various attributes of the project.

- **Exchanging Knowledge**: When the team members are distributed (e.g. the product owner is off-site) facilitating communication becomes an issue. *Physical meetings* improve the quality of electronic communication, and should therefore be considered. Improvement of technical tools is also recommended. Minimizing the number of intermediaries will have a positive effect on the*reciprocal knowledge sharing*.

- **Involving Stakeholders**: It is important that the organization is willing to make the necessary changes to support collaborating with the customer representative. Stakeholder involvement is one of these support functions. Raising awareness about agile among all stakeholders involved, and building a culture for collaborative and collective decision making between stakeholders and the team will help support the customer representative, as well as the developers.

8.3.1 Whom to Represent the Customer

After the organization has decided that customer involvement is the right way to go, the first step towards succeeding is choosing the right customer representative. This is a key success factor, and all alternatives deserve to be presented before the organization makes a choice. If the customer representative is to be the product owner at a Scrum team, Abrahamsson et al. (2002) recommend that this person would be chosen by the Scrum master, management and developers. However, as seen from the case study of Hanssen and Fægri (2006), letting sales and marketing pick the customer representative proved to be a good idea, since they knew the market and the customers. I believe that letting sales decide on who to represent the customer is a good idea, even if the customer representative should be in-house and not among the end-customers.

Whom to represent the customer depends on the need of the project, as well as the availability of suitable representatives. The representative will normally be one of the stakeholders of the project. I will here propose five stakeholders who are potential customer representatives: end customers, managers, developers and sales directors. The benefits and limitations of each of these are listed in Table 8.4. The most important decision is whether

Table 8.4: This table sums up the different alternatives for whom to play the part of the customer representatives, based on Sportradar's alternatives. Adjusting the precise benefits and drawbacks in that specific organization are likely to be needed.

Customer representative	**Benefits**	**Limitations**
End customers	Knows the customer domain in detail. Involvement can be used as a way of educating the customer in the system.	Non-technical, off-site, must be convinced to participate in the agile development. Communication with the developers can be a challenge due to different domain languages.
Sales directors	Knows the whole customer market, as well as having basic technical skills.	Busy schedule.
Managers	Partly technical, used to communicating with developers.	Normally not so familiar with the customer domain.
Developers	Technical, knows the possibilities and limitations of the system. Easiest form of communication towards the development team.	Has little knowledge about the customer domain, high chance of misunderstanding the customer requirements.

or not to use an external customer representative. If the domain is fully known by the developers or if domain knowledge is not required, there might be no need to bring in an external customer representative. On the other hand, if the system to be built is the first of its kind in that organization, and domain expertise is required, then having a panel of users might be the right way to gather requirements for the system.

The role of the customer representative is to provide the team with ongoing expertise. One of the major advantages of having an in-house customer representative is that it is easier to *guarantee* that the customer representative is available to the team. However, it is only feasible to have an in-house representative in those cases where the customer representative will be able to represent multiple customers and preferably also that the domain knowledge can be reused in other projects. If not, training an in-house representative would become too expensive.

The customer representative *does not need to be on-site and constantly available*, as especially XP requires. However, it is an advantage if the developers can contact the customer whenever they are stuck with a domain specific issue. As long as the dialog with the customer is relatively frequent, a part time representative can be sufficient for the developers to get the answers they require. This would make it more appealing for an end customer to provide a representative to the team, but this also opens up for having an in-house customer representative that simultaneously work with multiple teams.

In addition to *whom* to engage as the customer, it is also the question of *how many* customer voices that should be considered. When the market segment is diverse, more customer representatives increase the chance of eliciting all requirements. For example, at the Statistics project, employing one representative from both the media segment and the bookmaker segment would increase the chance of satisfying the needs of both segments simultaneously. The downside of multiple customer representatives is that there is a risk of running into communication obstacles (Pikkarainen et al., 2008). Many voices demanding to be heard at the same time can lead to confusion and disagreement, especially if the diversity in requirements from the customers is of great variety. Sportradar opted for making their projects flexible enough to support the functionality required by each market segment. If this is not an option, the company have to decide on who is the most important customer to please, which is a perfectly reasonable business decision.

The customer representative should have certain properties, for example communication skills and willingness to contribute. Finding a customer representative with a *technical background* is also an advantage (Karlström and

Runeson, 2006). As one of the managers in Trondheim pointed out:

> "It is easier to teach a technician the domain area rather that to teach a domain expert the technical aspects."
> — *Manger M1*

However, finding a customer representative with sophisticated domain expertise *and* sufficient technical skills for understanding the complexity and development challenges that lies in building a system might be the ideal, but seldom feasible. As the customer's main task is to provide the team with domain knowledge, the interface between technicians and non-technical customer representatives remain a challenge in agile development.

One of the risks of having a technical customer representative can be that this person takes over a too excessive part of the technical decision making. This is unfortunate, since the rest of the technical team can end up feeling less empowered, as the technical responsibilities shifts from the developers to the customer. The customer representative can also get a role resembling that of a *project leader*, which may interfere with the rest of the team members' sense of project ownership.

8.3.2 Exchanging Knowledge

Communication for knowledge sharing is a vital support functions when dealing with off-site customers and when collaborating in a physically distributed organization.

At Sportradar we have seen how certain divisions of the organization is somewhat isolated from other. For example, with few exceptions, there were little interaction between member of the upper management (including the product owner) and the developers. This can be explained partly by the lack of established communication channels between PO1 and the developers. All communication went through intermediaries causing the developers to think of "Gera as a black box, where information goes in and comes out, but no one knows exactly how it is processed within" — Developer D2. The developers did not have an overview of how the rest of the organization operated and therefore had a harder time understanding the main requirements and goal seen from the management's perspective. This *isolation of the team* is unfortunate because it can lead to wrong decisions being made at the implementation level.

Lack of communication prevent the developers and the customer representative to obtain a mutual understanding of the tasks at hand, which manifest itself as misunderstandings that at best lead to time-consuming confusions, and at worst lead to fatal design decisions that set the project back months, or even prevent the project from ever reaching completion. As an example, in section 8.2.3, I described situations where misunderstandings emerged from the lack of common terminology between the product owner and the developers.

Such situations as those Sportradar experienced can be prevented by building a mutual understanding of each other's domain language. Developers need insight in what the customers really need so that they can "feel" when they are doing something right, and when they can improve what they are doing further in order to give more value to the customer. And the other way, the customer representatives need to obtain an understanding of the resources required to perform the tasks they want done in order to better estimates.

There are three important steps that an team may take in order to obtain this mutual understanding: Minimizing the number of intermediaries, reciprocal knowledge sharing and

Minimizing the Number of Intermediaries

To support knowledge sharing, the number of intermediaries between end customers and developers should be kept to a minimum. Too long communication chains slows down the communication speed, but if the organization manage to shorten it, it can result in a more rapid accomplishments of goals (Melnik and Maurer, 2004). Another advantage of getting rid of intermediaries is clarification of goals, in the sense that the right person is being addressed. An illustration of this can be seen in the way the graphical designer was working:

> "It doesn't seem like we're making a product for the customer, but rather to our boss, because this German hierarchical system work like this: I have to sell *my* boss the idea, then he has to sell it to *his* boss and so on. We work towards pleasing our boss."
> — *Graphical designer GD2*

Working with the goal of pleasing one's supervisor was seen as a problem from GD2's view, since *answering to mangers takes the focus away from customer's requirements.*

While the number of intermediaries should be kept to a minimum, there are valid reasons for having them. For example, at Sportradar, managers were often intermediaries, filtering the information that went in and out of the agile team. Without any filtering, the channels between developers and customers would be flooded with information that possibly had no relevance to the developers and would only act as distractions.

When in-house customer representatives are used, they may act as filter for communication from the developers to the end customer. There are several reasons for an organization to chose this approach, most notably that representatives from end customers may feel more comfortable interacting with non-technical employees, such as the case with Sportnews.

In the situations where the in-house customer representative has in-depth domain knowledge, there may even be communication at all between developers and end customer since the customer representative is able to act as a stand-in for any of the projects end customers.

Reciprocal Knowledge Sharing

Reciprocal knowledge sharing is important for obtaining a mutual understanding between customers and developers, as well as establishing a common domain language. By letting the developers interact with the customer, either direct, through intermediaries or together with other stakeholders, knowledge is shared both ways.

The best way of facilitating reciprocal knowledge sharing is to allow developers direct access to the customer themselves, by having the developers visit the customer office, in order to observe and see how the product is being used.

If intermediaries between customer and developers are used as customer representative, it may be useful for developers to attend physical meetings together with the intermediaries and the end customer so that they can observe what concerns the end customers raise and gain a better understanding for the end customers requirements. After such meetings, developers should make sure they ask for clarifications on everything they did not fully understand.

> "No one at the Trondheim office has any knowledge about what the customer wants. I think it would have been a good idea to let some of the coders participate on sales meetings with the

customer, and to talk with the customer."
— *Manager M1*

Given that the customer representative is off-site, attending physical meetings may also make communication through IT-based channels such as email and instant messenger easier.

For organizations that operate largely within the same domain, developers will benefit even more from obtaining domain knowledge. For example, Sportradar operates within one domain, namely sports. No matter what the next project will be, chances are that it will have something to do with sports. Thus, domain knowledge obtained by developers from other previous projects can be reused in most new projects.

Effective communication

Together with culture and people, *communication is the most important success factor in agile software development* (Lindvall et al., 2002). At Sportradar, the most important communication channels are weekly videoconferences, company gathering and occasional physical meetings.

Physical meeting is an efficient way of exchanging ideas and knowledge compared to IT-based communication media, but has a cost related to the travel expenses and working hours of bringing intermediaries and the customer representative together with the developers when they are all located in different countries. Therefore, it is important that the agenda of these meetings are well prepared and as efficient as possible. At Sportradar, there were still room for improvement of the meeting schedules:

1. April 2009, PO1 came for a one day trip to Trondheim in, where he spent a majority of his time working alone in Redmine (see page 72).
2. September 2009, PO1 spent a week in Trondheim, mostly working on his own (see page 80).
3. April 2010, four sales directors came to visit the Trondheim office for one and a half day (see page 85).

To make the meetings more efficient, the organization can take use of *screencasts*[1] of the products to send out to the attendees prior to the meetings.

[1] A screencast is a digital recording of a computer screen output, often with audio describing the process.

This would make it redundant to explain the products at the meeting, thus saving time. The screencast might also make it easier for the sellers to understand how the product work, and it will give them a more visual sample to show their customers, in addition to documentation. Instead of using time on a walk through of all the products through a live demonstration, seller meetings, like the one in Vienna May 4 2010, could then be used to discus the content and value of the screencast.

As explained by Dennis and Valacich (1999), face-to-face communication is not necessarily the richest medium for communication—depending on what information that are to be transmitted, technical communication tools can be more suitable.

8.3.3 Involving Stakeholders

So far, I have mostly discussed the linear chain between end customers and developers. But projects often have other stakeholders that do not operate in this chain.

These stakeholders can offer various support functions and developers should consolidate these stakeholders in order to ensure that the requirements are well understood and that what they are doing conform with the management's policies.

The management in Trondheim also recognized the need for stakeholder feedback.

> "Since the product wasn't showed to anyone during the development, we are not sure if we have made something that the customer wants. The customers might have completely different requirements for the system that we have though of."
> — *Manager M1*

If the project make use of a customer proxy, the end customer is not actively participating in the product development. Still, occasional feedback from the customer may help confirming that the project is on the right track. It can also serve as a check to whether the voice of customer proxy reflects the opinions of the end customer.

An important note is that especially in agile development, developers are also important stakeholders whose opinions should not be left out. Conversely, developers ought to be encouraged to speak up and participate in meeting

and discussions with other stakeholders. Ignoring their competence will at best cause the organization to miss out on valuable feedback. At worst, the developers will feel unappreciated and thus less motivated and responsible for the outcome of the product.

As seen in Hanssen and Fægri (2006), the developers felt increased motivation and job satisfaction due constant reassuring from the customer that they were building something the right system. With this example one could argue that customer feedback is important, even if it just is for getter assurance that the project is on the right track.

Customers, developers and managers are not the only stakeholders that should be considered as participants in the project. As seen in the case study by Miller (2005), by using *interactions designers* as proxies for the development team, the companies under study managed to include many different voices into the development process, as the designers worked against a large spectrum of potential end users. Involving the right people who knows the customer domain is a reasonable plan for what stakeholders to involve. For example, at Sportradar, sales can be said to be such a group.

To improve stakeholder involvement, I have identified these initiatives that the organization should apply:

- Raising Awareness about Agile—Getting Management Support
- Offloading the Customer Representative

Raising Awareness about Agile—Getting Management Support

An important prerequisite to involving stakeholders of agile projects is that the stakeholders understand concepts of agile development.

All stakeholders and employees of an organization who have anything to do with the project should get an *introduction to the agile methodology applied*. This should be done even if the method is not intended to be implemented throughout the entire organization, as to raise awareness and understanding about the iterative, agile way of working. All stakeholders on an agile project should know about the core principles of agile, and which one that are applied in the organization. However, it is not necessary to educate each and every stakeholder in the agile manifesto; the principles applied are those that need to be presented.

The goal of informing the involved participants about agile practices is to give the organization and the stakeholders an understanding for how the

developers are working, and what their role as stakeholders is. That way, the stakeholders might recognize their opportunities to affect the development and the outcome of the project, motivating them to support and in best case to contribute.

In order to succeed with customer involvement in agile software development, *the agile team needs support from upper management*. This does not mean that an agile development team cannot coexist within the context of an otherwise plan driven organization. However, as we have seen at Sportradar, a plan driven management can be a hindrance for input and feedback from customers and stakeholders. Getting stakeholder input during the development process can prove difficult, since developers are not allowed to contact the end customer directly, as the management does not see the value of frequent interaction between coder and customer. Lack of support can also make it difficult for the stakeholders to "interrupt" the development process if they feel that the project is off track, as the management does not know that agile actually deals with these kinds of uncertainties and change in requirements.

In agile development, the software developers and the customer make most of the decisions. This requires a *culture for collaborative decision making*, supported by trust and respect among the employees. To build this culture, the organization have to be willing to spend an enormous amount of effort, time and patience (Nerur et al., 2005).

That being said, I do believe that Sportradar recognize the need for management support in agile development, based on the feedback from sales and management, who all agreed that they would have liked to contributed with feedback during development, and also letting the customer voice shine trough. But before Sportradar can allow for stakeholder feedback, the company culture must change. This can be done by providing courses or seminars explaining Scrum and agile to all the relevant employees of Sportradar, maybe even including the customer. Only that way will the management get an understanding for what the agile team need of resources and support.

Offloading the Customer Representative

Stakeholder involvement is not just important for development support, but also as a mens of supporting the customer representative. As seen from section 8.1.1, the roles of the customer representative can be challenging and wearing, due to high pressure, workload, conflicting roles and responsibilities. One way of offloading the customer representative is to involve more stakeholder in the development process, to ease some of the burden of the

customer representative.

Having one single customer representative is in itself a risk, in terms of getting the requirements right according to the whole customer market. Relying on one single voice to contribute with requirements on behalf of the customer is gambling from the organizations behalf.

In addition to having one customer representative, other stakeholder should be involved, so that the team is not dependent on one single person to provide domain knowledge into the requirements. As I have previously touched on, consolidating other stakeholder would be a good way to get confirmation that the project were on the right track even if the customer representative should provide sufficient domain knowledge.

Another advantage of having more stakeholders involved is the comfort in everyone knowing that there are agreement throughout the organization, or at least, that everyone who has something on their mind have the opportunity to speak up and to object if they disagree. *Collective responsibility* of the project makes it problematic to blame/reward one single person for the outcome of the project. Holding more stakeholders responsible for the project will offload the developers as well as the customer representative.

Chapter 9

Conclusion

With the introduction of agile mehtodologies, customer involvement has been put on the agenda and become a standard in the software development industry. Agile principles takes customer involvement one step further, and require a continuous integration of customer input and feedback throughout the whole development process.

Customer involvement in agile development usually works best when the customer representative is on-site, constantly available and fully dedicated. In reality, as we all know, the circumstances are not always perfect in the field of software development. As my study has showed, the business context does not always allow for an on-site, fully available customer representative, thus making the recommendations of how to involve the customer found in the agile literature difficult to accomplish. Constraints like time pressure, budget, efficiency requirements, intermediaries and bureaucracy, available resources and physically distributed employees are factors complicating the involvement of customer. When at the same time the customer representative has to be off-site and the management has little knowledge about the agile principles applied at the development team level, it creates a whole new set of challenges related to communication and facilitating reciprocal knowledge sharing between the customer and developers.

Motivations for practicing customer involvement are many, but so are the constraints. Thus, the question is no longer *if* we should practice customer involvement, but *how* and *to what extend*. It is also the dilemma of *who* to engage as the customer representative, a challenge yet to be addressed in the literature.

My suggestions for how to deal with these challenges has resulted in a *framework for practitioners*. In this framework, presented in section 8.3, I provide

a set of recommendations an organization should be aware of when practicing customer involvement in agile software development. The goal is to help the organization make conscious decision based on the awareness of all dilemmas and the opportunities available.

9.1 Implications for Practitioners

This thesis provides practitioners of customer involvement in agile software development with three implications:

- There is no one-size-fits-all solution to *whom to represent the customer*. Choosing the right customer representative is therefore required to be done on a per-project basis, depending on the constraints of the organization, the different alternatives for customer representatives available and the need of domain knowledge from the developers.

- When the team members are distributed and there are intermediaries between developers and customers, *facilitating communication* becomes an issue. Physical meetings between the various stakeholders improve the quality of electronic communication, and minimizing the number of intermediaries will have a positive effect on the reciprocal knowledge sharing.

- *Stakeholder involvement* is a vital support function for the agile project development to succeed. Raising awareness about the agile principles applied to the team among all stakeholders is recommended. Involvement of stakeholders have the advantage of supporting a culture of collaborative and collective decision making, offloading the customer representative, as well as the developers.

9.2 Implications for Researchers

In the literature there is a gap concerning how to practice customer involvement in an agile team when constraints makes unfeasible to apply an on-site, fully available customer representative. The question of who to engage has also been left unanswered. This thesis contributes with filling some of this gap, by proposing a set of what alternatives for customer representative(s)

to consider, and what their advantages and shortcomings are. I also contribute with a set of support functions that I seems to be needed due to the constraints, like stakeholder involvement and improvment of communication.

Still, there are a significant gap in the literature, and further research is required.

9.3 Further Research

The context of customer involvement is constantly changing, and due to globalization, acquisitions and multi-national organizations, the complexity and constraints put upon the organization increases. At the same time, the opportunities and options of how to relate to this context makes it more difficult to decide on a suitable practice for customer involvement. The software community is therefore in constant need of contributions based on expertize in the field of customer involvement. I suggest that more case studies is done in the field of customer involvement, addressing the constraints of having an off-site customer representative, intermediaries between developers and customer and a plan-driven management.

Along the course of my study, I encountered several questions that are yet to be answer in the literature.

- After having identified the candidates for the customer representative role, how are the different types of customer representatives best involved in the project? How does the engagement practice differ, regarding the different attributes of the customer representative?
- How should an organization involve the different stakeholders in agile development methods?
- When domain knowledge from the customer is to be transferred to the team, how is this done when the team consists of a relatively large number of developers? If it is not feasible to teach all the developers the customer's domain, which person on the team should be given the responsibility of learning the customers domain?

Bibliography

P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta. Agile software development methods. *Relatório Técnico, Finlândia*, 2002.

P. Abrahamsson, J. Warsta, M. Siponen, and J. Ronkainen. New directions on agile methods: a comparative analysis. In *Proceedings of the 25th International Conference on Software Engineering*, pages 244–254. IEEE Computer Society, 2003.

H. Aldrich and D. Herker. Boundary spanning roles and organizational structure. *The Academy of Management Review*, 2(2):217–230, 1977.

V. Basili and A. Turner. Iterative enhancement: A practical technique for software engineering. *IEEE Transactions on Software Engineering*, 1(4): 110–125, 1975.

K. Beck. Embracing change with extreme programming. *Computer*, 32(10): 70–77, 1999.

M. Beedle, M. Devos, Y. Sharon, K. Schwaber, and J. Sutherland. SCRUM: An extension pattern language for hyperproductive software development. *Pattern Languages of Program Design*, 4:637–651, 1999.

N. Bevan. International standards for HCI and usability. *International Journal of Human Computer Studies*, 55(4):533–552, 2001.

B. Boehm and R. Turner. *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley Professional, 2003a.

B. Boehm and R. Turner. People factors in software management: lessons from comparing agile and plan-driven methods. *Crosstalk-The Journal of Defense Software Engineering,(Dec 2003)*, 2003b.

B. Boehm and R. Turner. Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In *Proceedings of the 26th international Conference on Software Engineering*, page 719. IEEE Computer Society, 2004.

B. Boehm and R. Turner. Management challenges to implementing agile processes in traditional development organizations. *IEEE software*, pages 30–39, 2005.

B. W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21:61–72, 1988.

J. S. Brown and P. Duguid. Organizing knowledge. *California Management Review*, 40(3), 1998.

P. R. Carlile. A pragmatic view of knowledge and boundaries: Boundary objects in new product development. *Organization Science*, 13(4):442–455, 2002.

P. R. Carlile. Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science*, 15(5):555–568, 2004.

T. Chau and F. Maurer. Knowledge sharing in agile software teams. *Logic versus Approximation*, pages 173–183, 2004.

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

A. Cockburn. *Agile software development: the cooperative game*. Addison-Wesley, 2007.

A. Cockburn and J. Highsmith. Agile software development, the people factor. *Computer*, 34(11):131–133, 2001.

M. Cohn and D. Ford. Introducing an agile process to an organization. *Computer*, 36(6):74–78, 2003.

K. Conboy, X. Wang, and B. Fitzgerald. Creativity in Agile Systems Development: A Literature Review. *Information Systems–Creativity and Innovation in Small and Medium-Sized Enterprises*, pages 122–134, 2009.

L. Constantine. Process agility and software usability: Toward lightweight usage-centered design. *Information Age*, 8(8):1–10, 2002.

R. Daft and R. Lengel. Organizational information requirements, media richness and structural design. *Management science*, 32(5):554–571, 1986.

L. Damodaran. User involvement in the systems design process-a practical guide for users. *Behaviour & Information Technology*, 15(6):363–377, 1996.

R. Davies and L. Sedley. *Agile Coaching*. Pragmatic Bookshelf, 2009. ISBN 978-1-93435-643-2.

B. DeBrabander and A. Edstrom. Successful information system development projects. *Management Science*, 24(2):191–199, 1977.

A. Dennis and S. Kinney. Testing media richness theory in the new media: The effects of cues, feedback, and task equivocality. *Information Systems Research*, 9:256–274, 1998.

A. Dennis and J. Valacich. Rethinking media richness: Towards a theory of media synchronicity. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, volume 1. Citeseer, 1999.

M. Detweiler. Managing ucd within agile projects. *interactions*, 14(3):40–42, 2007.

J. Earthy, B. Jones, and N. Bevan. The improvement of human-centred processes-facing the challenge and reaping the benefit of ISO 13407. *International Journal of Human Computer Studies*, 55(4):553–586, 2001.

A. Eberlein and J. do Prado Leite. Agile requirements definition: A view from requirements engineering. In *International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Essen, Germany*. Citeseer, 2002.

M. Finsterwalder. Does xp need a professional customer? *XP2001: Workshop on Customer Involvement*, 2001.

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

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

M. Fowler and J. Highsmith. The agile manifesto. *Software Development*, 9 (8):28–35, 2001.

M. T. Hansen, N. Nohria, and T. Tierney. What's your strategy for managing knowledge. *Harvard Business Review*, 77(2):106–116, 1999.

G. K. Hanssen and T. E. Fægri. Agile customer engagement: a longitudinal qualitative case study. In *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 164–173, New York, NY, USA, 2006. ACM.

C. Hansson, Y. Dittrich, and D. Randall. Agile processes enhancing user participation for small providers of off-the-shelf software. *Extreme Programming and Agile Processes in Software Engineering*, pages 175–183, 2004.

J. Highsmith and A. Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001.

D. Hislop. *Knowledge Management In Organizations: A Critical Introduction*. Oxford University Press, second edition, 2009. ISBN 978-0-19-953497-5.

J. R. G. Jagdip Singh and G. K. Rhoads. Behavioral and psychological consequences of boundary spanning burnout for customer service representatives. *Journal of Marketing Research*, 31:558–569, 1994.

D. Kane. Finding a place for discount usability engineering in agile development: Throwing down the gauntlet. *Agile Development Conference/Australasian Database Conference*, 0:40–46, 2003.

D. Kane, M. Hohman, E. Cerami, M. McCormick, K. Kuhlmman, and J. Byrd. Agile methods in biomedical software development: a multi-site experience report. *Bmc Bioinformatics*, 7(1):273, 2006.

D. Karlström and P. Runeson. Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering*, 11(2):203–225, 2006.

M. Keil and E. Carmel. Customer-developer links in software development. *Commun. ACM*, 38(5):33–44, 1995.

H. Klein and M. Myers. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, pages 67–93, 1999.

M. Korkala, P. Abrahamsson, and P. Kyllonen. A case study on the impact of customer communication on defects in agile software development. In *AGILE '06: Proceedings of the conference on AGILE 2006*, pages 76–88. IEEE Computer Society, 2006.

J. Koskela and P. Abrahamsson. On-site customer in an xp project: Empirical results from a case study. *Software Process Improvement*, pages 1–11, 2004.

S. Kujala. User involvement: a review of the benefits and challenges. *Behaviour and Information Technology*, 22(1):1–16, 2003.

C. Larman and V. Basili. Iterative and incremental developments: A brief history. *Computer*, 36(6):47–56, 2003.

R. Leifer and A. Delbecq. Organizational/environmental interchange: A model of boundary spanning activity. *The Academy of Management Review*, 3(1):40–50, 1978.

N. Levina and E. Vaast. The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems,. *MIS Quarterly*, 29(2), 2005.

M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz. Empirical findings in agile methods. *Extreme Programming and Agile Methods-XP/Agile Universe 2002*, pages 81–92, 2002.

A. Lundkvist and A. Yakhlef. Customer involvement in new service development: a conversational approach. *Managing Service Quality*, 14(2/3): 249–257, 2004.

C. Mann and F. Maurer. A case study on the impact of scrum on overtime and customer satisfaction. In *ADC '05: Proceedings of the Agile Development Conference*, pages 70–79, Washington, DC, USA, 2005. IEEE Computer Society.

A. Martin, R. Biddle, and J. Noble. The xp customer role in practice: Three studies. *Agile Development Conference/Australasian Database Conference*, 0:42–54, 2004.

A. Martin, R. Biddle, and J. Noble. XP customer practices: A grounded theory. In *2009 Agile Conference*, pages 33–40. IEEE, 2009.

A. M. Martin. *The Role of Customers in Extreme Programming Projects*. PhD thesis, Victoria University of Wellington, 2009.

G. Melnik and F. Maurer. Direct verbal communication as a catalyst of agile knowledge sharing. *Agile Development Conference (ACD'04)*, 2004.

L. Miller. Case study of customer input for a successful product. In *ADC '05: Proceedings of the Agile Development Conference*, pages 225–234, Washington, DC, USA, 2005. IEEE Computer Society.

M. J. Muller and S. Kuhn. Participatory design. *Commun. ACM*, 36(6): 24–28, 1993.

S. Nerur and V. Balijepally. Theoretical reflections on agile development methodologies. *Communications of the ACM*, 50(3):83, 2007.

S. Nerur, R. Mahapatra, and G. Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):78, 2005.

J. Nielsen. Guerrilla HCI: Using discount usability engineering to penetrate the intimidation barrier. *Cost-justifying usability*, pages 245–272, 1994.

B. Oates. *Researching information systems and computing*. Sage Publications Ltd, 2006. ISBN 1-4129-0223-1.

F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, page 308. Citeseer, 2003.

D. L. Parnas and P. C. Clements. A rational design process: How and why to fake it. *IEEE Trans. Softw. Eng.*, 12(2):251–257, 1986.

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

D. Robey and D. Farrow. User involvement in information system development: A conflict model and empirical test. *Management Science*, 28(1): 73–85, 1982.

W. W. Royce. Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESTCON, Los Angeles*, pages 1–9, August 1970.

K. Schwaber et al. Scrum development process. In *OOPSLA Business Object Design and Implementation Workshop*, volume 27, pages 10–19. Citeseer, 1995.

C. Stevenson and A. Pols. An agile approach to a legacy system. *Extreme Programming and Agile Processes in Software Engineering*, pages 123–129, 2004.

M. L. Tushman and T. J. Scaland. Characteristics and external orientations of boundary spanning individuals. *The Academy of Management Journal*, 24(1):83–98, 1981a.

M. L. Tushman and T. J. Scaland. Boundary spanning individuals: Their role in information transfer and their antecedents. *The Academy of Magement Journal*, 24(2):289–305, 1981b.

G. Valkenhoef, T. Tervonen, B. Brock, and D. Postmus. Product and Release Planning Practices for Extreme Programming. *Agile Processes in Software Engineering and Extreme Programming*, pages 238–243, 2010.

L. Williams and A. Cockburn. Agile software development: It's about feedback and change. *IEEE Computer*, 36(6):39–43, June 2003.