NTNU – Trondheim
Norwegian University of
Science and Technology

An Investigation of Team Effectiveness in Agile Software Development

Lars Martin Riiser Haraldsen

Master of Science in Computer Science
Submission date: June 2012
Supervisor: Torgeir Dingsøyr, IDI

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

Abstract

Agile teamwork has been widely used and accepted in today's industry of software development. The methods in agile teamwork claim to improve performance and predictability, and has during the past years become the target for an emerging area of research. The majority of the existing studies concerning agile teamwork mainly focus around eXtreme Programming (XP).

This report is one of few that discuss teamwork in software development having the agile methodology "Scrum" in the main focus. The report focus on teamwork and team effectiveness. It discuss existing literature concerning Scrum and teamwork as well as showing the results from an observed ethnographically informed study of an agile project. All findings, challenges and opportunities, are analyzed and compared to theories around teamwork. The primary literature used are case studies about Scrum conducted in the past five years. Some descriptive literature is also used to support my findings.

My main results are that solid leadership and members willing to adapt are of great importance. I also found that working in a closed room together facilitates teamwork and can increase team effectivenes. My teamwork model and framework for this project might not be completely optimal for this specific observed project. Overall, I have found that Scrum can be hard to adapt to. However, the agile practices facilitates team effectiveness. In addition, my results show that Scrum guidelines support communication and adaptability.

Ultimately, it is interesting to see what can be improved in agile methods and to what extent team effectiveness changes.

Keywords: Agile development, Scrum, Teamwork, Team effectiveness, Team performance

Abbreviations

Abbreviation	Word/Meaning
PP	*Pair Programming*
XP	*eXtreme Programming*
SSO	*Single Sign On*
PB	*Product Backlog*
PBI	*Product Backlog Item*
SB	*Sprint Backlog*
QA	*Quality Assurance*
pdm	*Participation in decision making*
TfT	*Table for Tasks*
MPM	*Mutual Performance Monitoring*
CC	*Consultant Company*

Table 1: Abbreviations

Preface

This report is the result of a study conducted in my final semester of Computer Engineering at the Norwegian University of Technology (NTNU).

Ever since I first started working in projects back at secondary school, I have been intrigued by teamwork and observing how project members impact the results. When I was first introduced to Scrum I was thrilled by the idea of adapting it to the marked and the customer's needs. During my years at NTNU I have used Scrum in small projects, both student projects and summer-internship related, and found it to be effective. However, I wanted to find out more about what the success criterias are and how the methodology can be improved. I was determined to further explore the topic of agile development, having Scrum as main focus.

This report is an in-depth ethnographical study written by Lars Martin Riiser Haraldsen in January through June 2012. From the moment I started working with this master thesis, my intentions were to increase my knowledge of agile development and contribute with another case to existing literature.

Acknowledgements

I would like to use this opportunity to thank my advisor, Associate Professor Torgeir Dingsøyr for excellent guidance and knowledgeable feedback throughout the entire project. In addition, I would also like to thank the representatives from the observed case, both the consultant company and its client, for taking the time to let me interview and observe. The material gained from these interviews has been very informative and of great value.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software development and how it should be structured to produce faster and better solutions has been discussed for a long time. Working agile has become a "new" approach of creating software and has been adapted by the industry. Such methodologies claim to provide better predictability than traditional methods as well as a much improved team performance. This thesis is about software development teams using agile methods to organize their tasks and processes.

"Scrum is like chess"

Ken Schwaber (Scrum Originator)

This is a famous quote that describes Scrum and its process. Scrum is one of the most famous agile methodologies. Like chess, there are rules for Scrum which can be used straight forward. In chess you can follow the rules and move your pieces around randomly. However, to become a Grand Master in chess, you have to develop your pieces with a plan and purpose, creating a synergy that is strong enough to overpower your opponent. It is the same for agile development teams. To master and take full advantage of agile methodologies it requires more than following the given guidelines. This thesis will observe and discuss a team that has taken different agile methodologies and combined them to fit their working strategy the best.

1.1 Motivation and Focus

With this thesis I aim to gain a deeper understanding of agile teamwork and what makes it effective. Also I want to investigate if there is room for even

further improvements. Agile development has become more frequently used in the past decade and has become a hot topic in today's market. However, very little empirical data can be found on Scrum projects within software development. The number of existing scientific articles concerning Scrum is relatively low, and many of the published articles deals with lessons learned or adaption. Dybå and Dingsøyr [36]found 1996 studies on agile development, but only a single study addressed Scrum directly. Considering Scrum to be a potentially attractive methodology for a wide range of projects, I believe this area deserves further extensive research. I am greatful for the opertunity to observe and report from a live case. Thinking agile is a totally new way of thinking when it comes to team work. The focus has turned from a command-and-control view to a self-manageable team where everyone involved knows which areas the rest of the team works on as well. This creates opportunities where developers work in an environment where they get the resources and the help they need.

It is also interesting that most companies find the transition into agile teamwork difficult. This thesis will discuss a project that takes an experienced agile consultant company and mixes it with developers from a customer with no previous experience with the Scrum methodology.

I have analyzed existing literature on agile development and found to what extent it could affect team effectiveness. I wanted to study a project on my own to compare it and add to existing studies and to help this team become more effective. As there are especially few studies concerning Scrum, I found this area to be the most interesting.

1.2 Problem Definition

The results from this thesis seeks to add to the pool of existing Scrum studies. The approach is to observe a project where the Scrum methodology is used by a team where some of the workers are not familiar with the methodology. This is to determine whether or not Scrum facilitates team effectiveness and to discover the advantages and disadavantages of the method. The data I extract from the studied project will be compared to existing literature to illustrate how practice differs from theory in teamwork.

Considering the above, the main objective of this project was to investigate:

"How does the Scrum methodology facilitate team effectiveness and what

are the advantages and disadvantages?"

This beeing the primary research question, the focus of this report concerns Scrum methodology theory and how it facilitates teamwork in a real case. I want to find out whether or not the Scrum methodology is as good as the literature claims it to be and investigate wherethere could be room for improvement. This thesis investigates how a project team uses Scrum and adapts to it while trying to optimize their team effectiveness. The studied project uses principles from other agile methods as well. This will also be concidered and discussed in the report.

In regards to the research question it is relevant to investigate to what extent the Scrum methodology uses basic teamwork principles in order to produce team effectiveness. The model of Salas' "big five" will be used for this.

Having a self-organized team structure is what differ agile methodologies from traditional teams. This affects the leaders preferable behavior and how close he is working with the team. Additionally it is of interest to investigate how planning and progress are measured and carried out and how the software methodology has been adapted to fit the company's needs. It is believed that every project has different criterias for success and also different ways to complete these criterias. When it comes to Scrum it will be interesting to see how the project team adapts to the methodology and how this impacts the project. The adjustments a team makes for their specific project is crucial to the results of whether or not the team will work effectively and complete all requirements before the delivery date. This will be interesting to observe and investigate as well.

1.3 Report Scope

This report will describe an ethnograpically informed reasearch on a software development project. I was observing over half the projects period, and this report will present results according to team effectiveness. In addition, I have examined available existing literature in agile software development. The results here are analyzed and compared to the case research in order to add to existing research on teamwork.

1.4 Limitations and Restrictions

This depth study accounts for one semester, the tenth and final grade of my program of computer engineering. I have restricted the focus of my research to embrace Scrum as the primary teamwork model. However, there will be a short introduction to the meaning of Agile Software Development and some of its different team work models. This report will also go through the fundamentals of teamwork having Salas' teamwork model [Salas et al. 2005] as a starting point. I wanted to find out how well Scrum fits into this model. There are many interesting aspects, but having limited previous studies and time there are some areas I have chosen to cut out of this study:

- Scrum used in global software development. *Reason: Too narrow focus point.Working global is a totally different way of working in a team.*
- A thorough orientation of Scrum and its purpose. *Reason: I want my focus to be on the teamwork and not the Scrum guidelines. I will however present Scrum and its fundamentals.*
- A deeper look in the relations with the customer and the work concerning GUI and design discussions. *Reason: Too narrow. This is only a small part of Scrum.*

Primarily I wanted to focus on non-global Scrum teams where the team had to adapt to the Scrum methodology in some way. It is also important to point out that best practices can change through time. Thus, the results and conclusions of earlier researches might be outdated. This report have accounted for that by focusing on recent studies. Even though there are a large number of highly influential and important articles, newer studies will help describing today's development processes and teamwork more correctly.

1.5 Report Outline

The following chapters have different roles:

Background for Agile Development

This chapter will give the reader a solid background concerning agile methodologies and theories used in this thesis. The main focus will be Scrum.

Fundamentals in Teamwork

This chapter takes introduces the framework I have used to analyze my results. It presents a model that introduces core elements in succeding to make teamwork effective.

Research Method and Design

This chapter discusses the chosen literature used for this thesis. I present the retrieving process for previous articles and my chosen approach for the case study. It also explains how data was analyzed.

Results and Discussions

This chapter presents my results from the studied literature and the case study. The results are compared and discussed seperatly for each section.

Conclusion

This chapter will present my conclusion of this thesis.

Future Work

In this chapter I will suggest improvements and possible ways to take my work further.

Appendix A - eXtreme Programming

Gives a more thorough introduction to eXtreme Programming (XP).

Appendix B - Interview Guides

Shows the interview guides that are used for the case project.

Chapter 2

Background for Agile Development

This chapter will provide the reader with an understanding of the term "agile" and its meaning when it comes to Software Development. It is divided into sub-sections where each section discusses a used methodology within the studied project.

To understand what agile development really means, one can start with the definition of the word *"agile"*. A concise meaning of this word within the world of software development is hard to find. However, here is the definition from an online dictionary:http://dictionary.reference.com/browse/agile:

1. *Quick and well coordinated in movement*
2. *Active; lively: an agile person*
3. *Marked by an ability to think quickly; mentally acute or aware: She's 95 and still very agile.*

Take definition 1 and add "*software development*" and you get *"Quick and well coordinated in movement of software development"*. Quick and well coordinated can mean that the team is well coordinated. Adding definition 3 and you get a team that has the ability to think quickly and can change quickly. This is exactly what agile development is about; changing quickly according to the marked needs and the flow during development processes. Agile has also been defined in [18] where they look at many different studies concerning agile development.

"agility involves both the ability to adapt to different changes and to refine and fine-tune development processes as needed".

and

"agility "as the software team's capability to efficiently and effectively respond to and incorporate user requirement changes during the project life cycle."

are two definitions that explains agile development properly. If the reader of this report is not familiar with agile development I will advice him/her to read this article.

Early on, agile methods were first inspired by a trend in Japan after World War II [**?**]. The Japanese wanted to produce according to demand. This resulted in few wrong estimates and overtime work hours. Later, such methods were applied in software development and optimized over time. To work agile provides practical approaches which are summarized and stated as guiding principles in the Agile Manifesto: http://agilemanifesto.org/. The main values from this Manifesto are:

1. " ***Individuals and interactions*** over processes and tools"
2. " ***Working software*** over comprehensive documentation"
3. "***Customer collaboration*** over contract negotiation "
4. "***Responding to change*** over following a plan"

These values form the core of today's agile methodologies.

Project teams have during the past years showed an increase of interest in project teams [2]. This study has described the background philosophies concerning agile software development. It states that agile methods can be seen as a reaction plan to plan-based or traditional methods. In the study they describe agile development as:

"Methods for agile software development constitute a set of practices for software development that have been created by experienced practitioners."

In traditional methods it is believed that all problems are fully specifiable and that there allways exist a solution for each problem. Agile methods, however, address a more unpredictable world relying on people and their creativity. This thesis will not go any further into the difference between traditional and agile methods, however, I recommend reading[2] for further background information.

This study concludes that for small projects with changing requirements, agile methodologies work well..

Several different agile methodologies exist. Here is a list of the methodologies used in this project:

2.1 Extreme Programming

Extreme Programming (XP) focuses on best practice for development. From [2] they describe XP succinctly as: "

"Consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour weeks, on-site customers, and coding standards."

If the reader would like a more thorough review of XP take a look at Appendix A in chapter8.

2.2 Kanban

Kanban is another agile methodology for developing products and processes. The emphasis lies on delivering in time without working overtime. Kanban uses a task queue available to all participants where developers pull their work from.

The name Kanban originates from Japan where "Kan" means visual and "ban" means a board or card. Kanban was first used in a Toyota production system in the 1940's. The agile method itself is formulated by David Anderson and has three core principles:

- Start with what you know and what you do now. The Kanban method starts with what you have now and stimulates for further incremental changes to your system.
- Incremental and evolutionary change. The development team has to agree on that an incremental and evolutionary change is the way to improve the system.
- Respect each others responsibilities and roles. Every team has some elements that should be preserved. Kanban should be integrated among these processes and habits.

In David Anderson's *Kanban - Successful Evolutionary Change for your Technology Business* he describes five core properties. These are important for a successful implementation of the Kanban method:

1. *Visualize the workflow.* Visualizing the workflow is subtle. It is about revealing mechanisms: the interactions, the handoffs, the queue buffers, the waiting and the delays that are involved in the production of a piece of valuable software.

2. *Limit Work In Progress.* This implies the pull system which Kanban uses. The pull system acts as a stimuli for continuous incremental changes to the system.

3. *Measure and Manage Flow.* This point highlights the focus of keeping work moving and using the need for flow as a driver for improvement. The work should be monitored, measured and reported. By doing so, the system can be evaluated to have positive or negative effects on the team as a whole.

4. *Make Process Policies Explicit.* This point is to encourage the entire team and the leadership by reflecting the teams effectiveness. By having an explicit understanding it is possible to discuss in a more rational and empirical way. Thinking of a process as a set of policies rather than a workflow is a powerful technique.

5. *Use Models to Recognize Improvement Opportunities.* This shows that Kanban is quantitative and can take a scientific approach to improvements. When team members have a shared understanding of the work, workflow, processes and risks the effectiveness of the team will rise.

Kanban revolves around a board that is used to manage work in progress (hence the meaning of Kanban "Visual Board"). It is common for agile teams to put their user-stories on the board in different stages such as "in progress / development", "to testing" and "done". Such a board can be shown below in figure:2.1.

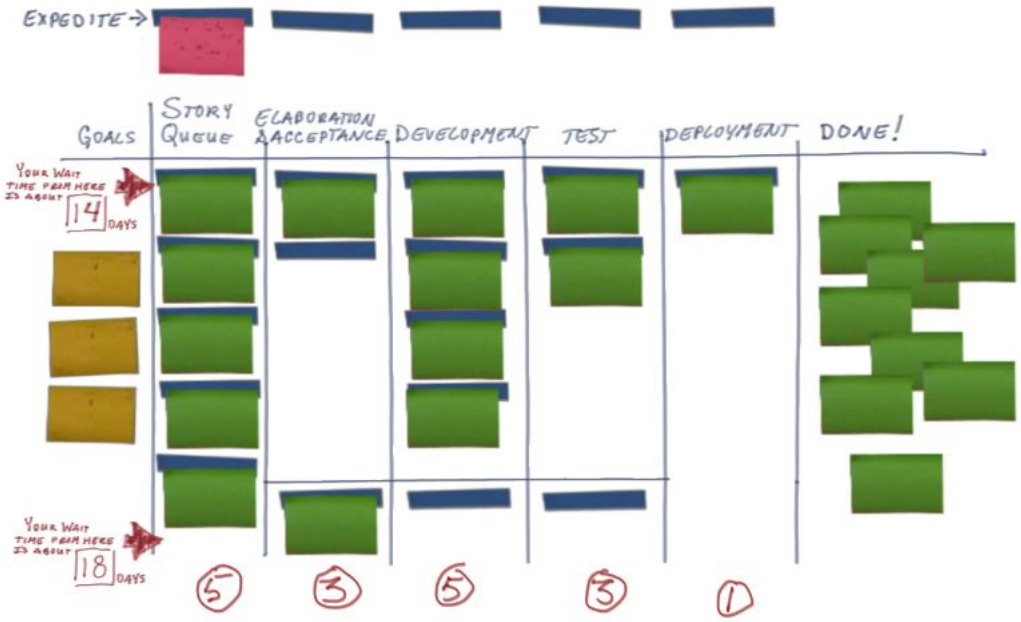

Figure 2.1: Kanban Board

The basic idea behind this board is that the user-stories travel from the left side of the board to the right. The board has seven columns. On the left side are the goals. These are kept on the left side so that everyone in the team knows what goals that are being worked on now. Right next to the goals column comes the story queue. This is a prioritized queue, where the top-most story will be the first to be started. The next four columns differs from team to team. On Figure 2.1 they have dividied them into different stages of the user-stories: Acceptance, Development, Test and Deployment. The last column shows the tasks when they are done and ready to be deployed into production.

Kanban enthusiasts claims that using such a method as the Kanban board can pay be highly benefitial [44]. They believe organizations that utilize this properly and leverage the high frequency of delivery will benefit financially.

2.3 Scrum

The word Scrum was actually first used in rugby as a name for a special strategy to get the ball back in to play. In Software Development it is an agile approach to manage projects [Abrahamsson et al., 2010, Linda Rising and Norman S. Janoff, 2000]. Scrum is designed to work empirical and should encapsulate existing practices. This, meaning that every individual scrum team for every individual project can combine scrum with other methodologies to fit the team and the project the best, thus maximizing team effectiveness.

In Scrum, processes are monitored and small adjustments to the requirements are made continously during the development phase. Scrum has a detailed supported guideline concerning the management aspects of a project, but lacks more details around the development itself. This sub chapter will provide the reader a short introduction to Scrum [Beedle et al., 2000] and its essential fundamentals.

Different roles

In a Scrum project the different stakeholders are dealt different roles. We have three different roles; the product owner, the Scrum master and the Scrum Team.

The product owner acts as a link between the Scrum team, the customer and all the other stakeholders of the project. His responsibility is to manage the product backlog.

The Scrum Master is responsible for the project following Scrum and its guidelines. His task is not to manage the project, but to work together with the team developing. His primary job is to facilitate the Scrum team so that the principles of Scrum are followed.

The Scrum Team is the core part. They are responsible for designing, developing, testing and often deploying the product. The team is self-organized and shares the same mental model and goals for the project. A typically ideal team consists of 5-10 persons.

Task Estimation

The task estimation is when the Scrum team works through the different requirements, breaks them up as much as possible and then estimates how many work hours each piece will take. There are several ways of doing this

and lots of tools that can help. An effective tool is called "Planning Poker" [Kjetil Moløkken-Østvold 2008]. In planning poker each developer is dealt a hand of cards where a hand presents all Fibonacci numbers from 1 to 100. The number presents work hours. For every piece of the work breakdown structure each developer draws a card from their hand and put it on the table. This card represents what he/her believes the time of implementation will take. The average time will be the estimate for this function. After each function is estimated all developers get his/her card back and they are ready to estimate the next function. This is done for each function/requirement in the project.

The Product Backlog

The product backlog is a list of all features or work that needs be completed. This list is prioritized based on customer needs and implementation time. New items can be added or deleted at any time. Such a document is needed before the implementation phase begins. However, the different features on the list are not binding. The product backlog also helps every developer get an overview of the system as a whole.

Sprint Backlog

The sprint backlog is a list of features taken from the product backlog. The amount of features taken depends on the length of the sprint, number of developers and how many work hours each feature is estimated to take. All of these features are decomposed to concrete tasks that are to be finished before the sprint is done.

Daily Meetings

Every work day the developers gather for a short meeting (around 15 minutes) to share three points:

- What has been completed since the last meeting?
- What will he/she be working on until the next meeting?
- Are there any issues that hinder progress in the development?

Usually these meetings take place at the same location to the same time every morning. This way, everybody has a concept of the general work and progress.

Sprint Planning and Review Meetings

The sprint planning meeting is a meeting prior to the sprint. It is used to create the sprint backlog. The result should be a plan that covers how the team should have a commitment ready at the end of the sprint. Such a commitment should include a working release of a part of the system. The sprint review meeting is a meeting performed at the end of each sprint. The product with its new implemented features are presented for the customer and other interested stakeholders. This enables good feedback from the customer and might provide new requirements or features to the product backlog.

Burndown Chart

The burndown chart is a tool very often used in Scrum projects. It shows the hours spent on each feature, all remaining hours and the working hours left of that sprint. It is useful as a management tool to monitor the working process and see whether adjustments are needed or not. Phases in a Scrum Project In a Scrum project there exists typically three phases: Pre-phase, development phase and a post-phase. The pre-phase consists of design and planning. In this phase, the product backlog is built along with short plan for the separate releases. This phase is typically as short as possible. The development phase is the part where the scrum team develops the product. It is separated into several sprints and shields the team from requirement changes and time-frames, as they are handled in between the sprints. This makes scrum agile and very flexible. The post-phase is after the last release. In this phase there are typically time for the last documentation, system testing and deployment. Scrum can be implemented in a project in several different ways. It is not always most beneficial to apply all standard “rules” when using Scrum either. It varies from project to project and team to team. Often Scrum is best combined with other agile methods.

Chapter 3

Team Effectiveness Model

There are several studies concerning teamwork and team effectiveness [43]. This report will focus on one model, but I strongly recommend the reader to take a closer look into [43], for further understanding of teamwork models.

According to James J. et al. [45] team effectiveness can be evaluated by using different objective or subjective measures. He states that objective measures usually miss critical factors. Because of this it appears to be agreement across studies that subjective measures, using scales combined of team satisfaction and ability to achieve goals to evalute team effectiveness. In this study I will define team effectiveness as a measure of a team working together in such a way that the core goals are accomplished within a certain time frame and the team members are satisfied.

To further categorize team effectiveness I will use a teamwork model from a solid study [**?**]. It will be described in the sections below. The model will be used to form the framework for the ethnographical study provided in chapter five.

Salas' study believes that good teamwork can be achieved if the right sets of components are there. In their study they discuss five core areas within a project teamwork and explains why each of these are important. The components discussed are: leadership, mutual performance monitoring, backup behavior, adaptability and team orientation. Their opinion is that all these areas are needed to achieve good team effectiveness, however, they can be implemented differently from project to project. This section will present the different areas and what is needed for each area to achieve good team effectiveness.

Below is a model presenting the the core areas. The model also shows

three coordinating mechanisms which support the areas to provide better performance.

The model is taken from [1]

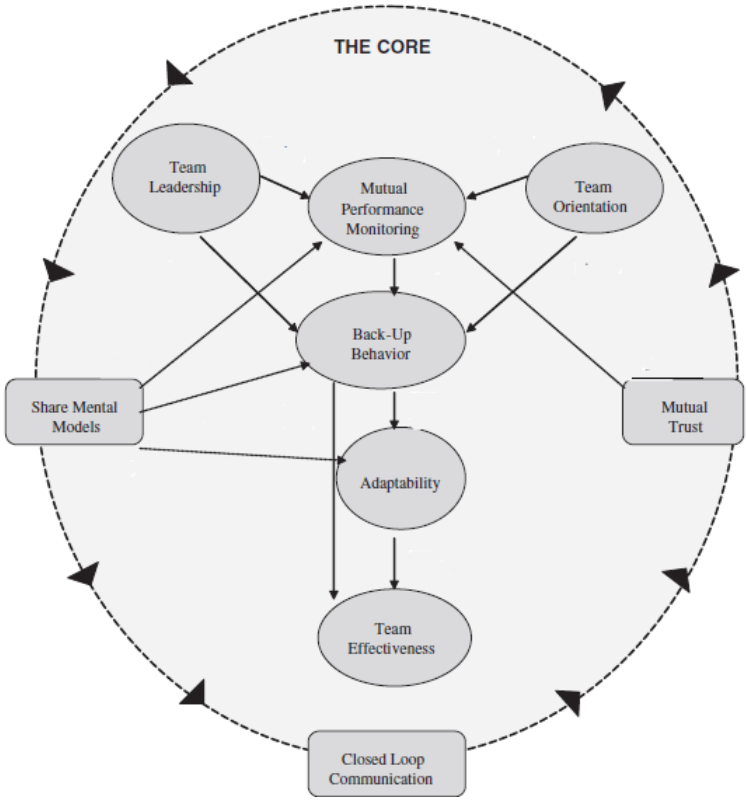

Figure 3.1: Salas' team effectiveness model. Taken from [1].

3.1 Leadership

Leadership is one of the core components of the model. It is discussed by lots and lots of studies where the majority concludes a team provides better with some sort of leadership. Without having a team leader that knows how to guide and facilitate profitably, team performance can decrease [6]. This is supported by Salas et al.'s studies where they state that a good team leader is key to maximize a teams performance abilities. The study explains that

a team leader should be able to generate possible solutions or at least start a thinking process within the team. He should be able to structure and organize the individual strengths and resources that each member provides, using the strengths to increase team performance.

A good leader has several tasks. Some of his tasks is to coordinate the processes, motivate the team members and provide the team with a shared mental model. A shared mental model is a mechanism that helps team members to interpret information the same way [**?**]. They state that such a model facilitates communication and coordination in a team setting. This information is also backed up by empirical evidence from [7] which states that:

"The provision of enriched information by team leaders results in more similar and accurate mental models among team leaders".

There are certain rules for how a leader should behave, mostly depending on how newly the team is established and their previous experience with working in teams. Typically will behavior, individual performance, motivation and trust change during a project, and the leader should change his leading style accordingly. In Durham et al. 1997 [**?**] they discuss two types of leadership. The first type includes a commanding leader, also known as the traditional leader from [10], which has a tough leadership style. The essential characteristic for this type is that he makes the decisions for the group. As a contrast to the commander or traditional leader, they discuss leadership within self-managing teams. Self-managing teams do not operate without a leader, but the role of their leaders has a completely different task. They are often called facilitators, coordinators or coaches and has as role to ensure good communication within the team. This type of leadership has been increasingly used in today's world of work and with great success [8]. In this study they issue the importance of participation in decision making (pdm) and how much a team can profit from it. They state that:

"*Participation by group members should be especially valuable when the task involves interdependence (i.e., it is a team task) and the team has to learn how to perform it effectively without prior training.*"

Traditional style of leadership would be likely to prevent or discourage such communication since their focus lies on getting subordinates to follow orders. They also advise that the team leader should develop team-based norms that enforce expectations between the different members of the team. When the team members feel they contribute to the team and understand the importance of working together team performance increased. This is

also backed up in [1, 7]. The results of Durhams et al.'s studies was that teams with coordinators tended to have higher performance than teams with commanders and that a more "participative" style was beneficial for the team.

3.2 Mutual Performance Monitoring

Mutual Performance Monitoring is the second core area Salas et al.'s studies look into. It has been defined in[11] as the ability to:

"keep track of fellow team members'work while carrying out their own . . . to ensure that everything is running as expected and . . . to ensure that they are following procedures correctly".

This study shows that effective teams have members that pay attention to what other team members do. This is especially important in stressful periods in the project, when team members potentially increase their mistake ratio. Having the knowledge of what fellow team members are working on makes it easier to encourage and facilitate each other. Studies show that individuals are often not aware of their own deficiencies when it comes to performance abilities. Getting feedback from a trusted team member can make individual aware of his or her deficiencies, thus increasing performance.

Salas et al.'s studies explains that keeping track of fellow team members are easier when they work close to each other. Working in the same room can provide the team easier access to understand who's working on the various tasks at all times. It also makes it easier for team member to help each other out.

As mentioned in 3.1 a shared mental model is also important to achieve effective performance monitoring. Such a model helps providing a shared understanding of the tasks, the goals and the team responsibilities. The model 3.1 shows that Mutual Performance Monitoring is connected with Mutual Trust. It is important that team members are open for feedback from other members. Trust within team settings has been defined as

"shared perception . . . that individuals in the team will perform particular actions important to its members and . . . will recognize and protect the rights and interests of all the team members engaged in their joint endeavor"[19].

Not having sufficient trust in other team members can expend time and energy in protecting, checking and inspecting each other instead of providing value-added ideas. Having team members feeling their input is not valued

or used appropriately they may be less willing to share the information [20]. The same study shows that team members might completely stop sharing information if they fear being perceived as incompetent. Trust is an important underlying mechanism to accept a certain amount of risk that comes when one have to rely on each other. Trust is a big part of Mutual Performance Monitoring considering that it is understood and accepted by team members that people are in fact looking out for each other for the good of the team.

3.3 Backup Behavior

Backup Behavior is the third area of Salas' model. It has also been described in [21], where he discuss three different situations where providing backup behavior is important:

1. To provide feedback and coaching to improve performance
2. To assist a team member in performing a task
3. To complete a task for a team member because he or she has an overloaded amount of work

The ability to reduce workload for an individual team member is important when it comes to team performance [1]. Salas' studies discuss that team effectiveness can increase when team members take over an already started task and finish it. They conclude that if the task of an overloaded team member is not facilitated or taken over, it is expected that team performance will degrade drastically. This is supported by [22] where they state that teams which are able to compensate for each other in periods of stress have fewer errors in their work.

The importance of backup behavior does not simply lie in the improved performance outcome, but also in how it affects team processes. Having good backup behavior allows for greater adaptability in changing situations and environments, thus increase team performance in such situations [21]. This can also be seen from 3.1 where backup behavior can lead to adaptability.

Salas' study discuss how a shared mental model is important in this area as well. Backup behavior is typically decided by the needs of the team. A shared mental model can form the core decisions of when a team member must step in and provide back up, how it should be done and what assistance is needed.

3.4 Adaptability

Adaptability is defined in [1] as the ability to understand and recognize deviations and then readjust accordingly. In projects there are almost never work that is performed 100% after plan and schedule. Typically a project consists of risks; information missing, new information emerging, team members become sick or important tools arrive later than planned. When such risks occur the team has to adapt to the new situation, and thus make adaptability a crucial area for team effectiveness. Research from [22] discuss this and concludes that teams being more adaptable were rated as more effective than others.

Adaptability assists teams to respond to unexpected demands. For adaptability to provide better team performance, the changes in the environment my constantly be concidered. Usually, when teams have worked on a project for a long time routines and habits have been established which can result in members not see changes in the environment as quickly. Salas' study states that such mindlessness can result in productivity loss or missed oppertunities for innovation and improvement. Sometimes it might be good to pause working for a short period of time just to overlook the project and work situation as its whole.

Team adaptability can be important for many different teams and in many different situations. It is important for team tasks that require innovation (e.g. research teams) or for teams that experience a failure (e.g. soccer teams that fail to achieve their goals e.g. winning the championship). Similar to backup behavior, team adaptability can be manifested in many ways depending on challenges and team tasks. The rate of work, how many tasks that is under progress and who performs the tasks are important factors that come into play when discussing adaptability. However, there are so many other factors that make adaptability very individual from team to team.

3.5 Team Orientation

The final aspect of Salas' model "The Big Five" is team orientation. Concidering the four previous areas to be more behavioral matters, team orientation is |more attitudinal. Team Orientation is the ability to take other team member's behavior into account and set team goals over individual goals. It is an important dimension within the "big five" as it improves both individual

effort and performance as well as individual satisfaction [22]. In addition it facilitates the overall team performance by for instance having better decision making within a team [23]. As they describe it in [23]:

"Team orientation is not only a preference for working with others but also a tendency to enhance individual performance through the coordination, evaluation, and utilization of task inputs from other members while performing group tasks".

Basically this means that each team member should aquire the skill to perform individually through input of other team members. This is of course in addition to the skill of working in a team. Salas' study states that individuals with a more team oriented view, would take other team members input when making a decision. Although the input was not always seen as correct, the quality of the decision improved.

In Salas' study they have also found that team orientation result in increased cooperation and coordination, which may increase task involvement, information sharing, strategizing and goal setting, thus improves team performance. This is backed up by many other studies: In [24] they have found that good team orientation results in increased cooperation and coordination among team members. In [23] they found that individuals with a team orientation more frequently considered teammate input to decide on a final course of action.

Chapter 4

Research Design

This chapter presents how I collected and chose the literature and my data from the empirical study that was used for this thesis. The work for this thesis has consisted of two primary ways of gathering data: Searching for studies from online databases and studying a project team using Scrum. This chapter will provide information concerning the study chosen, how data should be collected and its appropriateness. Section 4.1 will describe the collection of litterature studies and how I have used them for this thesis, while section 4.2 will describe the same for the ethnographical case study.

4.1 Literature Study

I wanted to search for studies in order to get an in-depth understanding of Scrum and how it is used in project management. This would allow me to have the necessary background information and be better prepared when observing the project team.

For the searching process I chose literature from a database called "Web of Science". The goal was to find around 20-30 different documents where I believed comparing approximately 8-10 would be sufficient. In the search for relevant and reliable articles my criterias were that:

- The article has to primarily be a Scrum study concerning a software development team that works in today's industry. This means that student projects and similar studies will be excluded. The majority of the article should be about scrum principles and its use in project management today.

- It has to be a scientific article. This means that experience reports and "lessons learned" - reports will be excluded.

- Documents focusing on the whole Scrum methodology. Many documents tend to focus on only seperate parts of Scrum in a project. I wanted to get as much data on the whole process as possible.

To fulfill my criterias I investigated all of the titles from the retrieved articles. If the article looked interesting, based on the title, I would examine the abstract and in some cases read parts of the article to further investigate if it was of any use.

The retrieving steps were:

- Refine my search to "only articles"

- Refine my search to "Computer Science Software Engineering" and "Computer Science Information Systems" articles.

- Sorting my search by "Times Cited"

- Removing duplicates shown in table 4.1 below.

Table4.1 shows my retrieving process presented in table. Table 4.2 then shows the titles of the articles found.

Table 4.1: Article Retrieving Steps

Keyword for Search	**Total Results**	**Total Results After Refining**	**Result After Choosing**
"Scrum Development"	251	25	6
"Agile Team"	672	62	6
"Agile Methodology"	841	80	3
"Scrum Software"	106	23	1

Table 4.2: Titles on retrieved articles

Keyword for Search	**Title on Articles**
"Scrum Development"	*"Customising agile methods to software practices at Intel Shannon",* *"A teamwork model for understanding an agile team: A case study of a Scrum project",* *"The impact of agile principles on market-driven software product development",* *"The agile requirements refinery: Applying SCRUM principles to software product management",* *"A decade of agile methodologies: Towards explaining agile software development",* *"Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges"*
"Agile Team"	*"Overcoming Barriers to Self-Management in Software Teams",* *"Agile Process Improvement: Diagnosis and Planning to Improve Teamwork",* *"Challenges to Teamwork: A Multiple Case Study of Two Agile Teams",* *"Overcoming Barriers to Self-Management in Software Teams",* *"Scrum and team effectiveness: Theory and practice",* *"Understanding self-organizing teams in agile software development"*
"Agile Methodology"	*"Beyond the customer: Opening the agile systems development process",* *"Agile Project Management: Steering fom the Edges",* *"The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects"*
"Scrum Software"	*"Software Development Methodologies, Agile Development and Usability Engineering"*

The table below, 4.3, shows each study that I have chosen to include as a basis for the literature study in this thesis.

Table 4.3: Articles retrieved and used for this thesis

Title	Author (s)	Published	ID
Customising agile methods to software practices at Intel Shannon	Brian Fitzgerald, Gerard Hartnett and Kieran Conboy	2006	[32]
A teamwork model for understanding an agile team: A case study of a Scrum project	Nils Brede Moe, Torgeir Dingsøyr and Tore Dybå	2010	[34]
Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges	Artem Marchenko and Pekka Abrahamsson	2008	[41]
The agile requirements refinery: Applying SCRUM principles to software product management	Kevin Vlaanderen, Slinger Jansen, Sjaak Brinkkemper and Erik Jaspers	2010	[9]
Overcoming Barriers to Self-Management in Software Teams	Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå	2009	[?]
Scrum and team effectiveness: Theory and practice	Nils Brede Moe and Torgeir Dingsøyr	2008	[3]
Understanding self-organizing teams in agile software development	Nils Brede Moe, Torgeir Dingsøyr, Tore Dybå	2008	[4]
Challenges to Teamwork: A Multiple Case Study of Two Agile Teams	Viktoria Gulliksen Stray, Nils Brede Moe, and Torgeir Dingsøyr	2012	[?]
Primavera Gets Agile: A Successful Transition to Agile Development	Bob Shatz and Ibrahim Abdelshafi	2005	[38]

As shown in table 4.3 nine studies have been included. The studies that are not included have been removed because it's either too short, did not

forfill my criterias or had too little focus on Scrum.

4.2 Ethnographically Informed Study

In [2]they state that there is a lack of quality Scrum studies. This concerns especially mature Scrum teams. Based on this and my motivation described in 1.1(ikke skrevet ferdig) I wanted to investigate what characterizes team effectiveness in Scrum teams.

4.2.1 Chosen Approach

There are several approaches to investigate teams on this matter. When studying teamwork within software development, case studies and ethnographically informed studies are the two most usual approaches. Based on a previous subject (IT3010 Research methods) I took this autumn and my preparatory project for this thesis I ended up doing an empirical ethnographical study. I believe my research question will be answered best by doing an such a study. Qualitative ethnographical studies are argued very positively in [37, 38]. Here he discuss researches concerning how projects are investigated with ethnographic research.

In [37], the study provides guidelines for how to investigate properly. He suggests that field notes should be written up regularly, interviews should be written as soon as possible and that the situations should be reviewed regularly during the research progress. The types of project I wanted to make research on can be seen similar to examples from [37, 38], which I also believe strengthen my choice of doing an ethnographically informed research.

Having this in mind I ended up with a project where a consulting company co-operated with their customer by forming a development team together. My case is more of a "convenience case" than a specific case to perform a specific study. The company and the educational institution allowed me to study this case and I was happy to do so because it was forfilled my criterias as well.

To increase validity and trustworthiness of my findings I used seve|ral ways of gathering data. My findings, results and data concerning the case project have been verified by the project leader from the consultant company. My basic approach to gather data:

- Observing meetings of interest. Such meetings are: Daily stand-up meetings, Sprint planning, Sprint retrospective, Demo's.
- Taking daily field notes
- Conducting interviews and "small talks"
- Looking into documents of importance. Such documents are: the burndown charts, the contract between the consultant company and the client, the product backlog, the sprint backlog.

The ethnograpical study was conducted by observing this project as an outside "researcher". By conducting interviews and having "small talks" with developers right after a discussion of interest, it was possible to record a representation of their daily work. I would also gather information from the contract and the product backlog to get a full understanding of the project. The burndown charts helped me monitor it and see whether or not the developers reached their sprint goals.

The structured interviews were conducted in the middle of the project and in the later phases. The interviewees worked either as Scrum Master or as a Project leader for either the consultant company or the customer. The small talks were conducted right after discussions or important meetings and the interviewees here were individuals from the development team. In total five interviews and four small talks were conducted. The small talks were semi-structured (depending of the conversation) with questions like:

1. "Can you tell me about the conversation you had with Person X"
2. "What were you thinking during this conversation?"
3. What do you think they/he/she were/was thinking ?"

However, the small-talks deviated a lot from this structure. All the interviews and small-talks have been transcribed and analyzed. The transcriptions have been sent back to the interviewees and been validated and accepted for me to use further in this thesis. The interview guides can be found in the appendix of this thesis. However, the interviews differed much from the original interview guides.

The developers from the consulting company are very experienced with Scrum and has been using it for many years. The developers from the customer, however, had not used it at all. The plan was that the customer's

developers would learn "as they work" through pair programming and guidance, with the idea that they could overtake the product and work on it by themseleves after delivery date.

4.2.2 Context

The studied project is a co-operative project between a consultant company and one of their customers. They call the project "Min Side" which is a student portal for information. The customer is an educational institution in Norway where the goal is to simplify information access and create a Single Sign On (SSO) portal for the students. The project owner believes it is important to emphasis that this is primarily the making of a student portal and that students will be the target that should benefit the most from this project. I was observing and interviewing the sprints from sprint 6 through 10.

Background

This project was initiated because a university wanted to optimize their communication flow. In 2009 the university established a project called "Front Office 2012". The customer issued Front Office 2012 because they wanted to improve their front end systems and improving communication. The project has as main goal to simplify a students' everyday life. The goal is that this result in less frequently asked questions for the employees - because the system is simplified! Having this solution should let the institution increase with bout 2000 students every year, without increasing staff. The product owner also believes that the project will increase communication and interaction with the students as well as become the primary way of information flow. It is divided into six different parts and the project I have studied for this master thesis is one of these parts.

General Information

The project I have been studying is called "My Page" and started 8th. of May 2011. The project completes with a working go-live version the 15th. of August 2012. The primary goal for this project is creating an SSO portal to professionalize their front end services and then creating "My Page" that provides all information a student need. The institution has approximately

18000 people that study and work there, and is growing by approximately 10-20% a year. They want to keep the same amount of administrative employees and need a system that allows that while the institution is growing. The system they use now does not allow that. It consists of different underlying systems which handle various areas from economics to individual and administrative tasks. The students do not find the information they are looking for, thus creating more administrative work. This is what the new system is supposed to prevent. The new system should include all underlying systems and feel like the user only runs one system. Basically the system takes an extra layer on top and gathers all services there. The system will handle functionalities like exams, mails, payments and information from the different courses and from the staff. Each of these functions were handled separately by underlying systems such as Aggresso, Banner, Its-learning among others.

In the beginning of this project the consultant company estimated the project to be 7300 working hours with nine sprints and an extra tenth buffer sprint. However this estimate was totally reevaluated already in Sprint two. Here they included 480 hours for the SSO and 120 hours for installation of QA test and development environments and ended up with an estimate of 8100 hours. Sprint 1-8 are used for developing and implementing. The 9th. sprint is a test sprint that involves two weeks of delivery tests and then two weeks of acceptance tests within the customers environment. Each sprint lasts for four weeks and has a capacity of 500 hours from the consultant company and 100 hours from the customer which result in a workload of 600 hours each sprint. {(8100-(10x600))=2100 timer til overs..?! (Spør prosjektledere!).}

- 480 timer har vært en del av prosjektet fra dag 1! Men det ble tatt inn som en del av backlogen - fordi disse timene hadde for mye usikkerhet. Disse ble tatt inn allerede i Sprint 0. Pga usikkerheten blir disse estimert etterværtog gjort om til pbi'er i hver sprint. POC - Sprint 0 - etter A/D fasen.

- Skriv om linje 3-6.

- Skille mellom utviklingsmetodikk og prosjekt. Selve prosjektet har: - LES Kontrakt og skriv om avnsitt!,

Below is a figure that illustrates a high level time line including project events and project satisfaction.

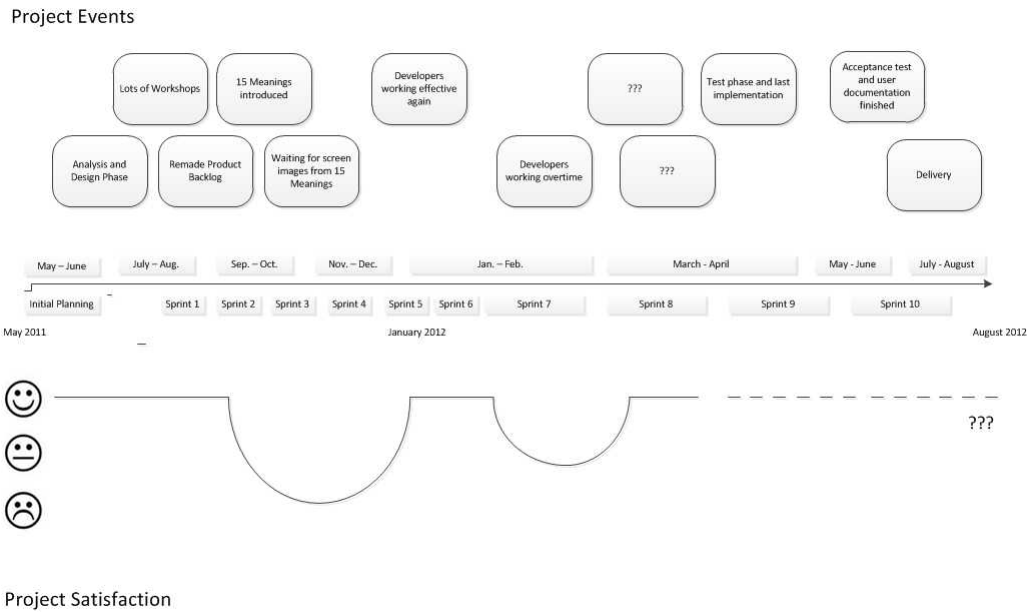

Figure 4.1: Project Events and Satisfaction

As shown in figure 4.1 the team has a very good overall satisfaction during most of the project. However, there were periods where not everybody was so satisfied and the graph drops down correspondingly. This will be more discussed in chapter 4.3.

Team Members, Roles and Elements

The development team itself consists of seven developers, where four of them come from the consultant company and three from the customer. The three developers from the customer work 25-50% of full work time at this project. The four developers from the consultant company work 60-100% on this project. Both sides have their own project leader. The development team uses a hybrid working approach where the combination of three different agile methodologies; Scrum, XP and Kanban creates the core principles for their working situation. Scrum covers the majority of this hybrid approach. The team members have very different knowledge with previous use of agile methodologies.The developers from the consultant company are very experienced with Scrum and the agile principles. They have had several other projects where Scrum has been used effectively. The developers from the

client's side are trying this style of work for the first time. One of the subgoals is to make them adapt to Scrum and enable them to take over the project for further developing and maintenance after version one is delivered.

The core of the Scrum/hybrid methodology, as viewed and changed by the project team, they have divided Scrum elements in four roles, five types of meetings and five types of artifacts. They are as follows:

Roles:

→ *Project Leader* - One for the vendor and one for the customer. Runs administrative and economical issues, overlooks the progress and controls time issues. The project leader on the customer side is also the *Project Owner* - a relevant term in Scrum.

→ *Scrum Master* - Is the subordinate and responsible for the final solution. He facilitates and drives the development team. He is solid and knowledgeable and he knows what is going on at all times.

→ Development *Team* - The team that develops and implements the solution. Scrum Master is also a part of this team.

→ *Functional Experts* - Experts on the system from the vendor's side of the project. They know how the components should look and work when the system is completed.

Meetings:

→ *Sprint Demo* - A demo presented by the development team, led by the Scrum Master, where the implemented components from the previous sprint are presented for the customer.

→ *Sprint Planning* - A planning meeting for the development team and often accompanied by a functional expert. Estimates and divides tasks for the upcoming sprint.

→ *Sprint Retrospective* - A meeting for the development team to go through pro's and con's for the previous sprint.

→ *Daily-standup Meetings* - Everyday meetings where the developers tell each other what they have been working on, what they will work on this day and if there are any problems that prevent their work. These meetings last approximately 10-15 minutes.The project leaders and functional experts are present at some of these meetings.

→ *Regular Meetings* - All other meetings. Typical if something out of the ordinary happens.

Artifacts:

→ *Product Backlog* - A list of items that is made from the customer's requirements specification.

→ *Sprint Backlog* - A list of items that should be implemented for a specific sprint.

→ *Burn-down Charts* - An administrative tool to see how much work that has been done and what remains for each sprint.

→ *Retrospective Table* - A table used in the process of Sprint Retrospective. Lists up all pro's and con's for a specific sprint.

→ *Table for Tasks in Progress (TfT), to QA, and for Documentation* - A table used to control the number of tasks in progress, in QA and for documentation.

Using this hybrid method and this setup provides what the team believes to give the best performance for this project. From XP they use pair programming to ensure that the customer's developers adapt to Scrum and can take care of the system after version one is released. The table for tasks in progress, QA and documentation is taken from the method Kanban and ensures that no more than 6 tasks should be under development, four tasks to documentation and two tasks in QA at the same time. Throughout the project, the sprint lengths of four weeks were kept constant and the team involved eight different developers, one scrum master and two project leaders.

Work Setting and Information Sharing

The development team works together in a closed room. Each developer primarily works individually on their own laptop connected to a large monitor. They have access to a projector and the TfT is visible for all to see. They use SharePoint to share documents and administer tasks.

The functional experts work in the room next door. This has changed several times during the project. At the beginning of the project, in the analysis and design phase, the functional experts and the developers worked closely together. After this phase they were separated and the functional experts worked in a completely different location. At that point the functionals were used more randomly. However, when they were moved to the room next to the developers, they were used more frequently.

4.2.3 Challenges

When doing ethnographic research I will face several challenges [F. J. Riemer, 2008, Article 12A]. I am aware that the team members and I will form some kind of relationship. They must all agree to participate in the research and fully understand the purpose of the research and the implications of their participation. Confidentiality is of course assured for all participants. I am also aware of the rigor within qualitative research and will describe the validity and reliability of my findings.

4.2.4 Bias

It is important to know my experience and limitations before starting such a study. At this moment I am a student at my fifth year in Computer Engineering. During my studies I have only used agile methods, and I thus believe that agile methodologies are more useful than the traditional ones. However, this can be negated because of my limited practical experience outside the university.

4.2.5 Data Analysis

In this section I will discuss what I have found from the chosen literature and the ethnographically informed study.

Interpreting data requires time. Because of the local research on one team the focus is deep.

"An ethnographer's job is to capture the thick description of an event, experience or scene" [38].

Because of the amount and variety of the collected data I chose a systematic and analytical approach to analyze the different data sufficiently. This goes for both the literature and the ethnographical results. Notes from interviews, burn down charts, product backlog and sprint backlogs and field notes are primary data sources that have been categorized and analyzed. The material are categorized in different folder: "Important documents", "Interviews" and "Field Notes". The table below 4.4 shows when I was collecting data for this study and what type of method I used the different dates.

Table 4.4: Data collection dates and methods for my case study

Date	Data Collection Method	Comments
27.01.2012	Interview	Interview with project leader from the consultant company. A thorough introduction to the project.
30.01.2012	Observation / field notes	Observed the demo and retrospective meeting of sprint 5 as well as planning meeting for sprint 6
31.01.2012	Observation / field notes and interview	Observed the daily working situation and interviewed the Scrum master.
22.02.2012	Observation / field notes, interview and small talks	I observed the daily working situation and interviewed the project leader from the customer's side. In addition, I found time to have conversations with two developers.
23.02.2012	Recieving burn-down charts updates	From this date and during the rest of the sprint I recieved daily updates of the burn-down charts.
27.02.2012	Observation / field notes and small talk	Observing the demo and retrospective meeting for sprint 6 as well as the planning meeting for sprint 7. Also got the chance to speak with one of the developers after the planning meeting.
28.03.2012	Observation / field notes and had a long talk with the project leader from the consultant company	Clarified the context and corrected all the mistakes concerning the project part of my thesis.

Table 4.5: Data collection dates and methods for my case study, part 2

Date	Data Collection Method	Comments
29.03.2012	Observation / field notes	I observed the daily working situation and got an interview with one of the developers.
Easter 2012	Studied contract, change order papers, requirement specification papers and the product backlog	Reading... Lots of reading.
18.04.2012	Observation / field notes and small talk	Observing the daily working situation and had small talks with the functionals.
19.04.2012	Observation / field notes	Observing the daily working situation
27.04.2012	Observation / field notes	Observing the daily working situation
03.05.2012	Skype interview	Interview with the project leader from the consultant company
May 2012	Communication through email and phone	Communication with Scrum master and the project leader from the consultant company

Having the different data in the different folders I further categorized my analyzed data. Concidering the framework that was established in chapter 3 I categorized my findings based on the focus areas of the model 3.1 from Salas' studies. There are five main areas: Leadership, Mutual Performance Monitoring, Backup Behaviour, Adaptability and Team Orientation. These five areas form the core of the analysis done for this thesis. Some data that were collected are concerned with the side elements from the model: Mutual Trust and Shared Mental Model. These will be pointed out in the report, but still connected to one of the five main areas. The same categories are used for the literature studies.

4.3 Important Data Collection

This chapter will present the most important ways I observed to collect data. It focus on how the different meetings were conducted. This chapter will also focus on how problems were facilitated within my studied case. Solving problems are crucial and often connected with *three* main areas of Scrum; Sprint Planning, Sprint Retrspective and Daily-standup Meetings. The observed team had, however, very few complications. This chapter will explain how the observed team used these aspects of Scrum to handle both external and internal demands and issues.

4.3.1 Sprint Planning

Sprint planning meetings are an important part of Scrum. Every iteration of Scrum begins with this meeting and is supposed to be a conversation between the product owner and the development team. At this meeting there should be an agreement on which tasks have the highest priority and thus which should be implemented first.

For the observed team the meetings were led by the Scrum master and took place before each sprint. The meetings were consequently after the retrospective meetings and always on the same day. For the observed sprints, the observed team diversed from the original Scrum guidelines. The tasks from the product backlog were given priority and estimated. The Scrum master would have examined the product backlog and explained all tasks to the team so that everybody knew how much work each task involved. If the Scrum master does not know exactly how a function should work, a functional worker is called in. A functional worker is an employee at the educational institution and is working (and will be working after the project is finished) with these tasks. At these meetings there is often a functional present to help explaining the functionality of certain tasks. In this way the team gets an understanding of all tasks and know what should be done for the upcomming sprint. After a task is explained properly, the team will do their own estimates and then allocate the task to one of the developers. This deviates from the original Scrum principles. In this way the developers know from the beginning of the sprint what they are expected to complete at the end of each sprint and can plan their upcomming four weeks. The Scrum master remarked that:

"Allocating tasks like these were an adaptment we made for the team and

it has worked out nicely for us. Up to around sprint three we only allocated the first task and then finished that before adding a new task to each developer. Now the developers can choose what they want to work with right away."

Consequently, the more enthusiastic and competent developers chose the preferred tasks, while the more passive developers ended up with more "boring", but often easier tasks. The difference of enthusiasm and competence within the team was a large gap. To close this gap experiential learning and pairprogramming were used. Some tasks were given to two programmers that should pair up; one skilled and one that should learn. This could prove to be difficult because all team members worked with different time capacity on the project. The developers that worked only 20%-50% on the project could not be assigned difficult tasks, because the competence and time at that point would not make them able to finish the task. They were instead given easier tasks and help so they could learn and improve during the project. All tasks were added to burndown charts and updated every day by the project leader.

The scrum master shows a very good understanding of the system as a whole, as well as what is good for the team. He does most of the talking at these meetings, however, there are discussions under the estimation process. He appears as a knowledgeable and technical developer as well as an extrovert person who is good with people. Because of his technical expertise the project leaders give him absolute credibility so that he can decide which tasks should be implemented at what point.

4.3.2 Sprint Retrospective

Sprint Retrospective meetings are also a core part of Scrum. They are held at the end of each sprint and involve a discussion between the scrum master and the team where the themes are pro's and con's of the finished sprint. The meeting is held so that the team can learn and adapt from mistakes and/or positive incidents, thus increasing team effectiveness of future sprints.

For the observed team, the retrospective meeting was held right before the sprint planning having the project leader present. The meetings are led by the project leader and the scrum master. The team use a few minutes where each member writes down what went well on a post-it note. These notes are placed on a board and the team members also explain why this specific incident was positive for them. After each team member has placed at least one post-it on the board, the scrum master groups them together (because there are often similarities) and each team member will agree on

what are the most important points. These are written down and will be rememberd for future sprints. After a round of pro's, the same thing is done for con's. The team become aware of their deficiencies and thus improves effectiveness of future sprints.

In one of the retrospective meetings the several team members pointed out issues where they could not get the screen images from the external company that designs them fast enough. When these images were not delivered to a certain time, the developers were set back and had to focus on other tasks.

4.3.3 Daily-standup Meetings

Daily-standup meetings are are meetings happening every day at the same time and place. They should not last for more than 10-15 minutes and every developer should answer three questions: What have I done since last meeting? What am I suppose to work with today? What problems have occured? Having answered these questions while all team members are listening will give everyone some understanding of what individual developer has been working on.

For the observed sprints, the team had these meeting every day, except for the days when retrospective and sprint planning meetings took plce. The project leaders were present some of these daily-standup meetings to follow up and get an understanding whether or not the team is doing what they should. These meetings were also used to discuss issues the development team experienced. Usually such discussions were stopped and continued after the daily-standup with only the concerned people.

4.3.4 Use of Burn-down Charts

Burn-down charts are graphical representations of work left versus time left. In ths project they were used actively. The project leader updated them every day such that the development team could see what had been done and what was left to do. Whenever a task was completed, the developer completing this task would flag it "done" and write up ("burn down") the number of hours spent on this task. This would be visualized on the next burn-down chart. The charts were also very useful to see how many hours that had to be moved to the next sprint.

Chapter 5

Results and Discussions

In this chapter I will present what I have learned from studying the literature and the case. I wanted to find an answer to:

"How does the Scrum methodology facilitate team effectiveness and what are the advantages and disadvantages?"

This chapter will focus on the teamwork model and discuss my findings for each section to answer this research question. I have categorized the results in tables with an emphasis on Salas' model 3.1 and this chapter is structured accordingly. The following tables will show what areas in Salas' model the results belong to. For each section I will first present the findings before presenting and elaborating the results from the case study. The results will then be discussed. This type of structure has become more and more popular in ethnographically informed studies. This way of structuring also makes it easier to contextualize my findings. Some of the results will occur in more than one table as they touch on several focus areas.

For the literature results tables each row contains one finding with a corresponding result. In addition I also show an ID which is a reference to the literature where this finding originates. The findings from my observed case are shown in the case study tables. These results have been given an ID with a "#" in front (like this: #X) to seperate them from the literature results. The case tables also present how I gathered the data for the specific finding. The ID's will be used when discussing the results.

The literatures used are mostly case studies that are conducted over the years 2008-2010. The studies consist of small Scrum teams (4-8 team members). The contents focus on the working process and situations where team members express themselves regarding to what extent they believe Scrum is

working for them.

5.1 Leadership

Leadership in Scrum is a shared responsibility shared between the team members. Seeing as Scrum is meant for self-managing teams, the Scrum master will in most cases work as a type of leader the rest of the team can turn to for assitance and guidance. Salas et al.[1] describes how leadership impacts team effectiveness as:

"Leadership affects team effectiveness not by handling down solutions to the team, but rather by facilitating team problem solving through cognitive processes, coordination processes, and the team's collective motivation and behaviors."

Leadership has been studied and discussed by lots of researchers. My theory section 3.1 emphasizes findings from [1] where facilitation and coordinating processes and communication are the main focus.

Below are tables presenting the results within the leadership category.

Table 5.1: Literature Results - Leadership

ID	Findings	Results
[38]	Hired a coach to facilitate the beginning phase of Scrum. The study shows that having a coach that acts as a temporary leader helps the team familiarize itself with the guidelines of Scrum and thus use it effectively at an early stage.	Team effectiveness increased
[40]	Using burn-down charts to monitor the team members and making these charts visible for the team members. The results gave workers a feeling that their work was meaningful. It was also a great way for the project leader to monitor the work process.	Team effectiveness increases
[35]	A finding that concerned people not listening to each other when discussing technical issues. From the study the developer state: *"When we discuss technical issues, it often ends in a kind of ''religious" discussion, and then I give up. And then you let people continue do what they are doing."* This is related to leadership concidering everyone in the team shuld be heard.	Developers stopped participating in debates, less feedback were given, motivation decreased, and thus effectivenes possibly decreased as well.
[35]	The team suffering from developers not following the provided guidelines. The Scrum Master was not experienced and did not manage to lead the meetings properly.	Team effectiveness decreases
[34]	Transition was difficult and slow, resulting in the Scrum Master taking more control than Scrum guidelines allow. The Scrum Master proceeded in the same fashion in which he had run the entire project and did not adapt when things were not optimal.	Team effectiveness decreases

Table 5.2: Case Results - Leadership

ID	Collection Method	Findings	Results
#1	Observation and Interviews	The Scrum Master is extremely valuable and a great asset to the project. He is solid at a high technical and social level and has complete control of everything that goes on at all times. He has a clear vision of results, goals and tasks that are under progress and tasks that have not been started. He also shows a good understanding of which developers are best suited for the different tasks.	Team effectiveness is solid and the project has a stable progression.
#2	Observation	The developers from the consultant company were experienced with Scrum.	This facilitates the self-managing and leadership of Scrum.
#3	Interviews, burn-down charts	Good planning between the project leaders. Meetings are scheduled and carried out as planned. Extra resources are allocated right away when necessary.	Ensures a stable progression in the project.
#4	Observation and interviews	The team had no problems with self-organization and communicated well with both the Scrum Master and the project leaders.	Facilitates good communication and helps to improve teamwork.
#5	Observation from retro-spective and planning meetings	The team achieves a mutual understanding of results, tasks and goals. At these meetings the Scrum Master presents and explains each task to ensure everyone understands what each tasks involves.	Facilititates a shared mental model and a common understanding of the goals for the upcomming sprint.

Table 5.3: Case Results - Leadership2

ID	Collection Method	Findings	Results
#6	Observation from Demo	The Scrum Master leads and presents their work on the demo. Each developer, however, also has to prepare a part that they have worked and present it individually.	This facilitates better understanding of the project as its whole and can contribute in a slow but steady increase of effectiveness.
#7	Interviews	Towards the end of the project, the QA environment collapsed. This incident would set back the team if it had not been for good decision-making by the scrum master and the project leader. An individual server: a "stand-alone box"was made.	An important function of the system had to be delayed. However, the project could continue as planned because this function was given to external developers from America.
#8	Burn-down charts	Updated burn-down charts were sent to all team members every morning.	Facilitates motivation and makes team members feel they contribute to the project
#9	Observation from planning meetings	Scrum-Master was talking so much that it seemed hard for the "weaker" programmers to join the discussion at all.	Does not correspond with Scrum guidelines.

Leading a team can prove to be difficult. The literature results from [35, 34] show how difficult it can be to lead a self-managing team. Who should take responsibility to guide and facilitate when the team struggles? In Scrum, leadership is a shared responsibility between all the roles. A leader should be able to direct, guide and coordinate other team members, assign tasks and establish a general positive atmosphere among other things. In Scrum, the aspect of teamwork is a fine balance between intervention and motivation for the Scrum-Master. In the literature [38] they hired a coach to

facilitate Scrum in the beginning of the project. This worked out positively and team effectiveness increased. In my observed case, however, hireing a coach was not necesarry and therefor not done. The Scrum master and the developers from the consultant company were experienced in using Scrum [Result #2] and could act as coaches toward the inexperienced developers. From interviews and observation I learned that the learning process was slow and that a lot of time was consumed by it. This could, however, be for many reasons. It could also be that the less experienced developers would not easily adapt to the Scrum process while all the experienced developers were present. Maybe they would rather use Scrum more efficiently when trying completely on their own in future events.

Within my case, leadership worked relatively well without any important issues concerning team effectiveness. One of my findings is that "The team had no problems with self-organization and communicated well with both the Scrum master and the project leaders" [Result #3]. This can be seen from observation and interviews. One of the developers states that:

"I think scrum master is a great guy! He is socially intelligent and technically very smart. I believe I speak for the whole team when I say that communication with him works very well."

This is backed up by observations from the daily work as well. The scrum master excels as a remarkable and solid leader with great technical knowledge as well as beeing ambulatory at a social level [Result #1]. From observations I noticed that close to all technical issues was primarily discussed with him. If any developer wondered about something or had any technical issues, the scrum master would be the first person the team member would ask.

The Scrum Master's central role within the team could also be observed from the different meetings. When the project leader was not present, the Scrum master was the one leading these meetings. That the scrum-master contributes and appears as a good role model is one of the reasons why this project works so well. This is backed up by theory in [1] which they state that:

"a leaders contribution to the team is one of the key points to increasing team performance."

In section 3.1 it is presented that a team leader should be able to generate possible solutions or at least start a thinking process within the team. From my results I would say that Scrum Master would give solutions to the team in many ways. This could be seen from observation where developers are

stuck with a task. If the problem is not handled and discussed right away, it is discussed under a daily-standup meeting.

The retrospective and planning meetings were in the form of "present and discuss" where discussions within the team were emphasized. The Scrum master was usually the one doing most of the talking and the discussions were usually between the consultant company's developers. A result of this was that the other developers had a hard time joining the discussions [Result #8]. A developer said:

"Sometimes it can be hard to enter the discussions. Especially if its not within my area. I don't really mind him talking this much. He knows this better than anyone else here so why shouldn't he?"

.In my literature studies [35] one can see a similarity *where* the person leading a discussion not listening to the other team members. In the literature studies this resulted in less team effectiveness and is also backed up from theory [1] where it is discussed the importance of listening to each team member. In the case study, however, there seemed to be several reasons why people did not participate equally in the discussions. Reasons such as: not so experienced in the topics, lack of interest and that the developers did not work 100% on this project are the most important ones. That beeing said, I do not believe team effectiveness was reduced in my observed case, as it was in the literature and in theory. Having a Scrum-master pushing the discussion forward and dividing tasks as they were, seemed to be the most optimized way of doing things [Result #5]. This also made the team gain a common understanding for what needed to be done. Such a shared mental model and a common understanding for the upcomming sprint facilitates good teamwork. This is also backed up in theory from Salas' model as can be seen in chapter3.1.

From both the literature result [40] and results from my case, burn-down charts were used actively and the results were positive [Result #7]. These charts helped the developers to see to what extent they were scheduled with the tasks and how much that remained this sprint. The chart below from 5.1is taken from the middle of sprint 8 and shows that the team is pretty much in schedule. The blue area covers the tasks that has not been started on yet and as can be seen from the figure, these tasks follows the guided black diagonal line pretty nicely.

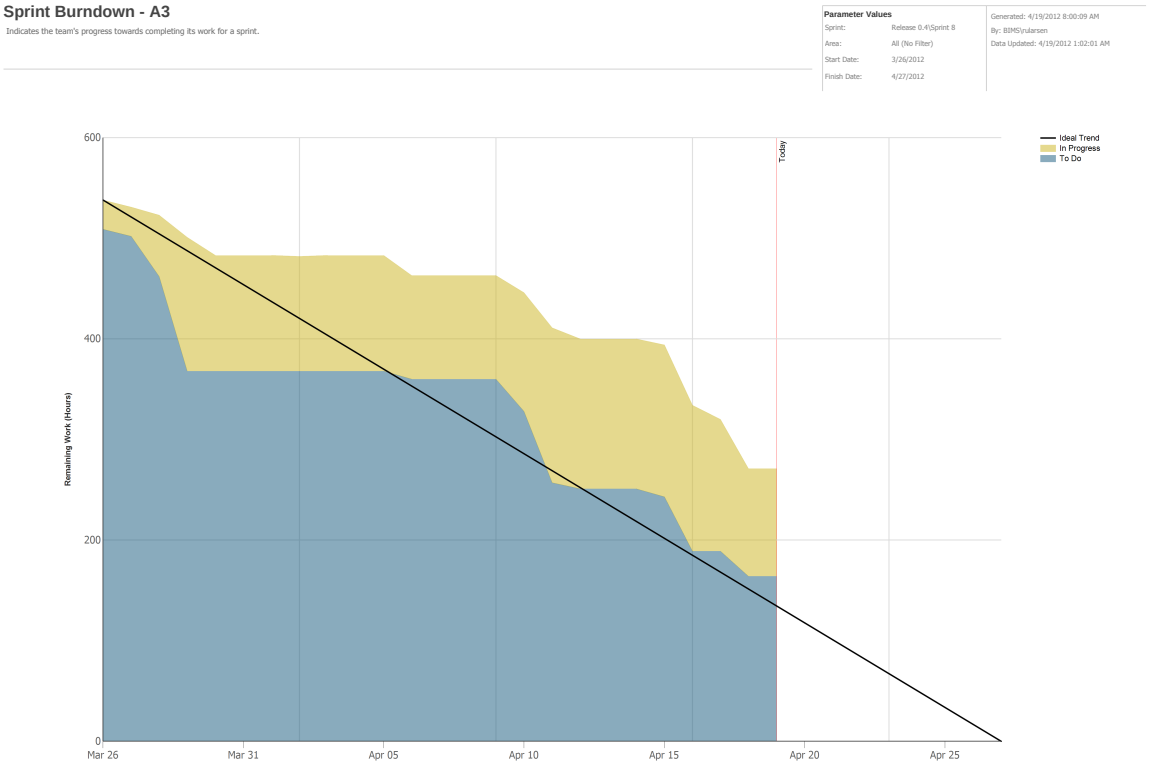

Figure 5.1: Burn-down chart from sprint 8

That managers use management tools such as burn-down charts is a good way to monitor and control the work progress. This type of tool also shows the work effort of the team together and make the developers feel they contribute to the project. This is also confirmed in the theory [21, 1] where it is stated that a team leader facilitates team effectiveness by monitoring the internal and external environment.

The theory in [1] describes how a leader should combine and synchronize the individual contribution of each team member. In my observed case, the developers with less experience are set to do tasks they are capable of completing. The team also use pair programming - comparing a weaker developer with a stronger one which facilitates the individual contribution to the team.

5.2 Mutual Performance Monitoring

In section 3.2MPM is defined as:

"keep track of fellow team members'work while carrying out their own . . . to ensure that everything is running as expected and . . . to ensure that they are following procedures correctly. "

MPM emphasizes working in the same room, having a shared mental model and information sharing among other things. Below I will present and discuss my findings from this area.

My results are presented in the tables below:

Table 5.4: Literature Results - Mutual Performance Monitoring

ID	Findings	Results
[41]	This project had a developer working overindividually (see (*) further down in this table). After a serious conversation with the rest of team he started attending the daily meetings and participated more in team related discussions.	The team environment became more productive and collaborative
[42]	Developers complained about a lack of information about the progress of the project. *"The information flow has been low since the beginning of this project ten years ago."*	Developers cared less and less about the project.
[41]	A developers worked individually. Not attending daily-standup meetings and not knowing what the others were working on. The developer in this specific case was a specialist, but implemented results that poorly integrated with the rest of the team. (*)	Frustration among team members. Work had to be redone. Not efficient!
[3]	The team became unaware of what the others were working on and thereby lost sight of the big picture.	Daily standups became uninteresting. Team effectiveness decreased

Table 5.5: Case Results - Mutual Performance Monitoring

ID	Collection Method	Findings	Results
#1	Observation	The location setting of the developers working environment is a closed room with the functionals in the neighbour room. However, only the consultant company's consultants were seated here together.	Facilitates MPM for half the development team.
#2	Burn-down charts	Burn-down charts were used for monitoring performance. They are updated each day so that all team members can see how the team is doing.	Facilitates MPM.
#3	Interview and observation	Scrum master facilitates developers working on similar tasks to talk to each other to save time.	The developers are co-operating, but it does not seem to save any time.
#4	Interviews and observation	From an interview with the scrum master and two of the developers it was clear that the team members did not know what other members were working on.	Positive and negative. The developers seems more focused on their own tasks. However, the downside is that they cannot take advantage of other peoples work when working on similar tasks.

Mutual performance monitoring is facilitated through several aspects of Scrum. The daily scrum meetings, the review, planning and retrospective meetings and the burndown charts are all elements where mutual performance monitoring is featured. Theory from Salas[1] states that communication is key for this area. In the literature[41] I found that some developers did not attend the daily meetings, but working completely on their own. They created good solutions for their tasks, but they were poorly integrated with the rest of the developers. This resulted in extra hours spent on reprogram-

ming and team effectiveness was obviously decreased. It was not the case in the project I observed. Here, developers were allways present at the daily-standups (if nothing special had occured). The project leaders also attended these meetings when they could find the time.

The developers from the consultant company shared a closed room [Result #1]. Below are some photos of their working station:

Figure 5.2: From the work station of the development team

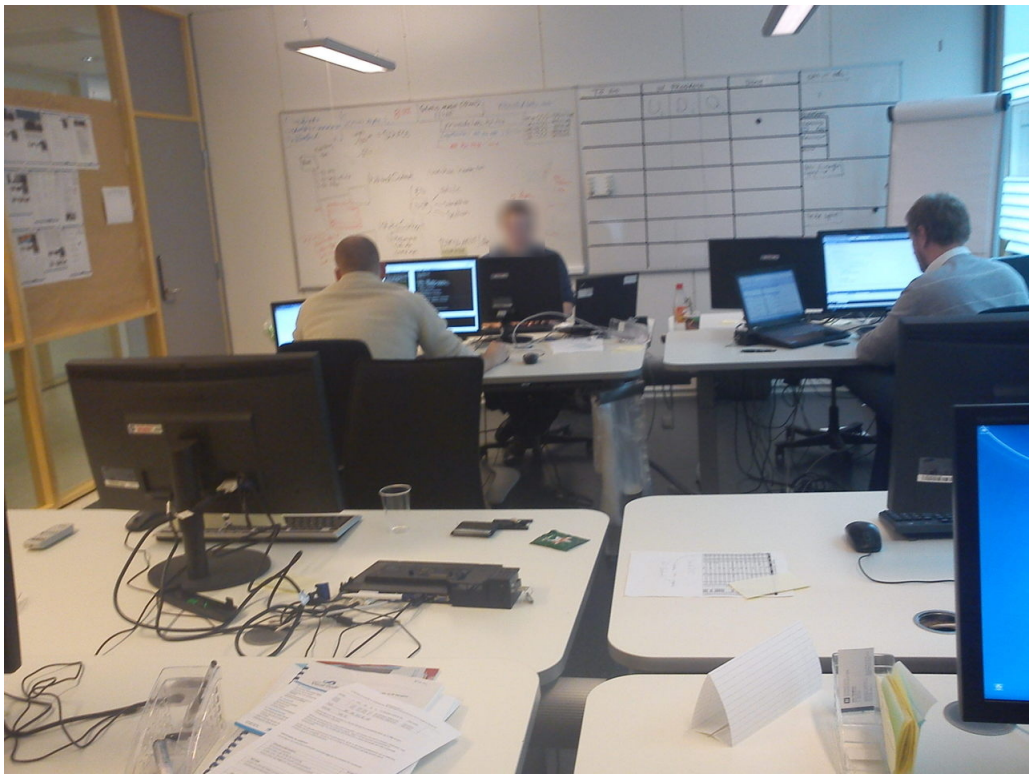

Figure 5.3: From the workstation with the Kanban board in the background

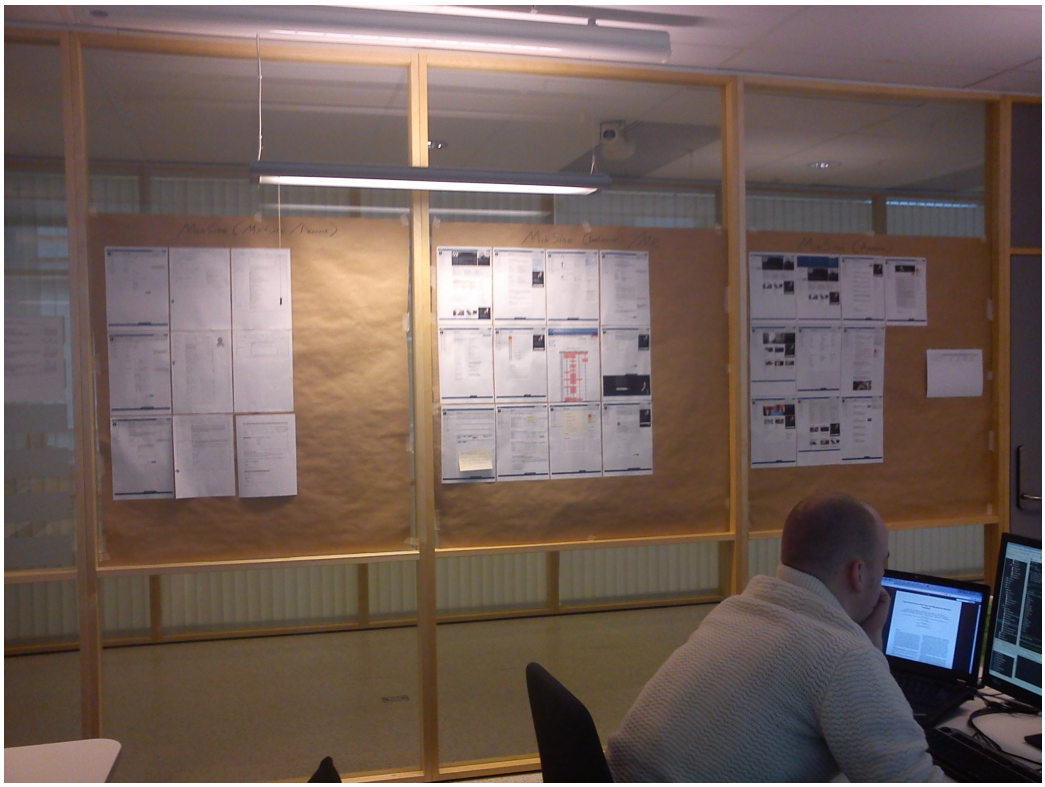

Figure 5.4: Project Work Station Wall

Concidering how the developers are situated there exist both positive and negative sides. As can be seen from the photos, the developers are sitting in a closed room where it is easy to monitor and communicate. However, only half the team is working in this closed room. This facilitates mutual performance monitoring for these developers. Having the developers seated like this improves communication and enables more feedback. It also facilitates the social parts of a project where the developers more often have lunch together. However, as for the other half, this is decreased - at least to a certain extent. I would state only to a certain extent, becaue whenever the developers are pair programming this happens in the same room. This pair programming facilitated a learning process for the customer's consultants. One of the developers from the customer side said:

"I really enjoy working two and two. I learn a lot from this process!"

However it did not seem to save any time later in the project [Result #3]. A reason can be that the customer's consultants in this room are only

dedicated 20%-60% of their time to this project and thus the learning process took very long time.

Another postive thing about this room is that the functionel workers are seaeted in the room next door [Result #1]. This has worked out good for the team. From observation I see that the functionals are more used and the feedbacks concerning the screen images are given more frequently.

In [3] they found that developers complained about not knowing what the different members were working on. The developers did not feel they could see "the whole picture" and that this decreased the productiveness. In my observed case I would argue that all developers do not know what the others are working with. Especially the developers seated in a different room do not know what all the others are working on [Result #4]. However, I do not believe this decreased team effectiveness, because the developers working 100% of the project are seated together and know more or less what is going on.

As mentioned in chapter 5.1 burn-down charts also facilitates mutual performance monitoring [Result #2]. The employers can pay attention to what has been done and what remains of work. The project leader states:

"We use the burn-down charts actively. This is because the developers can easily see how far they have come with their work for this sprint, and its a good tool for me to monitor the work process. This is excellent for us!"

From observations I can see that the burn-down charts are actively used. However, it is hard to see whether or not team effectiveness was increased. I would have to compare this to other projects to see if it really helped team effectiveness or not.

5.3 Backup Behavior

Within the Scrum guidelines, there are no direct practices that supports backup behavior. However, as described in section 3.3, this behavior is of big importance when it comes to team effectiveness. While Scrum relies on beeing an effective self organizing team, backup behavior also concerns providing feedback, coaching and facilitating and assitance from team members to complete tasks for each other.

Below I will present and discuss my findings related to backup behavior.

The tables below show my results for backup behavior.

Table 5.6: Literature Results - Backup Behavior

ID	Findings	Results
[42]	Too many specialists were working on the same project. Nobody in the team could help out with when a specialist became overloaded with work. The specialist states: *"We clearly have trouble transferring knowledge. The project is so big that everyone cannot know everything."*	Tasks were completed later than scheduled.
[42]	No backup behavior was ensured. The developers did not care whether they finished 90% or 70% of their work at each sprint. The remaining work was simply moved to the next sprint.	Working overtime and an "isolation" period where the team members were not allowed incoming phone calls.
[41]	Overindividualism was found in this study. In the case studied two developers were "free" from the daily-standups unless they had something they wanted to share.	One of the developers produced work that was poorly integrated with the rest of the team

Table 5.7: Case Results - Backup Behavior

ID	Collection Method	Findings	Results
#1	Interviews and observation	One week, some of the developers had to work overtime to achieve the sprint goals. Some developers were overloaded with work and received help from fellow team workers.	This facilitates backup behavior. It helped the team to catch up and achieve the sprint goals.
#2	Interview	Towards the end of the project, the QA environment collapsed. This resulted in external developers being hired to implement one of the important functions.	The project could continue as planned.
#3	Interviews, burn-down charts	Good planning between the project leaders. Meetings are scheduled and carried out as planned. Extra resources are allocated right away when necessary.	Ensures a stable progression in the project.
#4	Observation	There were incidents where developers from the client's side needed help finishing their tasks. More experienced developers came to assist as soon as they had a hand free.	This facilitate good backup behavior and better team effectiveness.
#5	Observation from retrospective	Developers would speak out when they had issues with some of the tasks or happenings in the last sprint.	The developers would get help with the task. Facilitates backup behavior
#6	Observation during demo	The project leader from the client side as well as the functionals gave good feedback on the displays and functions that were presented	The development team knows what needed a "fix" in the upcomming sprint.

Backup behavior was one of the areas with the least findings and results. This area might need more research to get good results. In my literature findings the [42] they faced a challenge where too many specialists were working on the same project. Nobody else could handle their work and thus, when these developers became overloaded, tasks could not be completed as sched-

uled. The developers would not care if tasks for the given sprint were finished. In this case all unfinished tasks were simply moved to the next sprint. Working like this violates with point three from section 3.3:

"To complete a task for a team member because he or she has an overloaded amount of work"

and results in reduced team effectiveness. In the same study, I could see that individual goals were given a higher priority compared to the team goals. The developers chose tasks that looked interesting prior to the highly prioritized tasks. This project ended with an "isolation" period where all the developers were "isolated" in a room till the project was finished. This is obviously not a wanted position, concidering working overtime is one of the primary issues that should disappaer when transitioning into working agile.

In my observed case, backup behavior was not a problem. Mutual communication between the leaders, the scrum master and the development team worked well in the project [Result #2, #3 and #5]. An example can be the organized weekly meetings between scrum master and project leaders. Also, the daily-standup meetings and retrospective meetings showed that the team members would speak out whenever problems occured. From a regular work day I observed a developer said:

"... I don't think I can handle this task on my own ..."

It did not take long before the scrum master came and assisted him [Result #4]. Additional incidents similar to this were observed, with and without the scrum master interfering. A few times when a developer needed help the scrum master arranged for pair programming with a developer with more expertise in the needed area.

There was a small period where the developers worked overtime [#Result 1]. This lasted for approximately one week. Agile methods do not emphasize working overtime. However, I believe the development team solved this period perfectly. The overloaded workers recieved assitance from workers with some extra time. The willingness to work and helping out facilitates backup behaiour and helped the team out, so they did not fall behind the next sprint. Such willingness to work is researched and backed up by theory to increase team effectiveness [1].

The developers showed evidence of effective backup behavior through pair programming, assistance and feedback given [Result #4 and #6].

A developer said:

"Pair programming has worked excellently. I appreciate this and think it helps us learning this way of working as well as knowledge transfer concerning

programming."

From observation and interviews it was easy to see that backup behavior was facilitated both from the developers and the leaders. Because of good planning and agile decision making, external resources were initiated when needed [#Result 3]. Also, when developers had difficulties in completing their tasks, developers with more expertise within that area were helping out. This clearly improved team efficiency for this project.

5.4 Adaptability

Adaptability is probably the area within Salas' model that profit most from agile methodologies. In section 3.4 I haved defined adaptability as

"the ability to understand and recognize deviations and then readjust accordingly".

Scrum facilitates the ability to readjust to deviations. This is because of the constant flow of feedback and communication between the development team and the customer. Adaptability can be important for many different teams and in many different ways and can be the big difference between failure and success.

The tables below present my results related to adaptability.

Table 5.8: Literature Results - Adaptability

ID	Findings	Results
[40]	Pair programming provided the team twith good knowledge sharing and were used on specific difficult tasks.	The weaker developers improved and the team as a whole became more adaptable over time. Lead to a smaller code base. Team effectiveness slowly increased.
[42]	The client approached the developers directly and said: *"We have to do something about this now!"* - referring to a specific tasks. The developers then faced the problem of going past the limit for maximum number of tasks they are supposed to have in progress.	Team members said yes to the client's demands but were not able to adapt properly to handle them. This resulted in tasks beeing delayed.
[41]	The Scrum-Master tended to be "too concerned" about solving people's problems instead of allowing them to solve the problems themselves. As a result the team decided not to let the Scrum Master be present at team retrospective meetings.	The team found the retrospectives to be more comfortable. However, the retrospectives became less organized and also less effective.

Table 5.9: Literature results - adaptability part 2

ID	Findings	Results
[41]	The developers lacked interest in adapting to the agile principles. The Scrum master from one of the cases studied in this article stated: *"it becomes quite easy to drift back to the "old ways" when nobody takes personal interest in Agile"*	The team experienced a long period where nobody took initiative to lead the agile principles properly. This resulted in people not always knowing what to do, which in turn made them less effective.
[3]	Customer support interrupted the work of the developers.	The team did not manage to adapt to this. The result was that the Scrum Master took charge of the direction and the team became less self-managing. Team effectiveness decreased
[41]	The team faced difficulties when adapting to Scrum. The Scrum Master placed an over-emphasis on Scrum and the team did not share all of this emphasis.	Developers spent time to argue whether or not Scrum was suitable for them and therefore lost time developing.

Table 5.10: Case results - adaptability

ID	Collection Method	Findings	Results
#1	Interview and observation	The Scrum Master sets a "weaker" programmer together with a "stronger" programmer to make them work together in pair-programming. From interviews and observations of retrospective and planning meetings the developers are very satisfied with this way of working.	Facilitates both adaptability for the weaker programmer as well as good team orientation. However, hours dedicated to this type of work means less hours focusing 100% on developing.
#2	Interview and observation	One week, some of the developers had to work overtime to achieve the sprint goals. None of them seemed unhappy about this, but rather supported the process and wanted to complete the tasks.	This shows good adaptability and desire to work. The developers adapted to the extra workload and managed to overcome it.
#3	Interview	There was a challenge with the screen images which was given late to the developers.	Developers had to work on other aspects of the project, but were probably not so efficient as they could have been if the screen images were ready.

Table 5.11: Case results - adaptability part 2

ID	Collection Method	Findings	Results
#4	Interview	There was a change order in the requirements specification which dedicated more work to the deveopers.	Extra resources were initiated and the developers allready working in the project dedicated more time to the project. Showed good adaptability!
#5	Interview	There was a challenge with the test environment beeing down.	The project leader, in discussion with scrum master made some quick decisions and took care of it. Showed good adaptability!
#6	Interviews	Adapting to Scrum became a challenge. One of the reasons might have been not having a big design up front. Some developers started questioning whether this was the best way of working in this project	Some of the developers struggled adapting to Scrum.

From the theory [1] they discuss that almost no software development projects are performed 100% as planned and scheduled. This is also shown in both my findings from the literature [41, 3] and in my case study [Result #4]. Throughout the project, adaptabilty concerning the tasks were quite effective. There was a challenge where the customer added extra functionality to the requirements specification. In this case extra resources were added and more time was dedicated for some developers.

During the project there were few technical problems. For instance, towards the end of the project, the testing environment being down become an issue [Result #5]. The leaders and the team had to adapt to the situation. They discussed various solutions for the problem, but ended up with a quick fix that worked out for all parts: Creating a stand-alone box, a test-server

for the different operators where testing was needed. The project leader announced in an interview:

"The last update is that QA is down. ....we can still be on schedule, but there has to be some changes.... I have discussed with scrum master and we ended up with creating a test-server where this is needed"

This fix helped the team to be on schedule and continue as planned.

In the literature [41] most teams struggle with adapting to scrum. There can be several reasons for this. In[41] the scrum master had so much emphasis on scrum that developers started questioning whether the methodology was good for the team or not. In my studied case the adapting process was also a struggle. From interviews I learned that most developers wanted to try Scrum and was set to do so. It is seen from theory, literature[41] and in this case [Result #6] that adapting to agile methods are difficult. In my studied case there are team members who find the traditional way better. To facilitate the adaption to agile methods they used pair programming [Result #1]. From observation I learned that the learning process was still slow, but the developers seemed more happy with the situation.

From an interview with the project leader from the client's side I found that he believed in a more long-term planning and that Scrum did not fullfill this to the wanted expectations. He explains that if the development tasks would be 100% clear at all times Scrum would probably have worked better. He has a point concidering a period where the screen images were not delievered to the developers on time [Result #3]. A better approach to this could be that the external company which designs these images could have been included earlier in the project, maybe even before the development phase. This does not mean the developers did not deliver anything in this phase, but it was an unfortunate situation where efficiency was slightly decreased. The project leader states:

"The company handling screen images should have been initiated earlier. Possibly in the design phase. Having them working a few months this autumn before the developers were initiated would benefit the project. I also believe this would help us having developed more in this phase as well."

That some developers struggle adapting to Scrum can also be seen from observation. I believe that developers have too little time dedicated to this project. Having other paralell ongoing projects that require just as much attention affects the agile principles for this project. The developers not familiar with Scrum is working 20%-60% on this project and this affects adapting properly. From interviews and observation I could see that one of

the developers not familiar with Scrum before the project, had adapted and begun to like Scrum and its way of working. He enjoys the project so much that he gladly work overtime which gives him about 100% on this project. This tells me that adapting to a new methodology also concerns interest for the method and the project. Obviously interest for a project will increase the individual efficiency, but it can also affect the entire team positively. When he spends this much time on the project it is also easier to become familiar and adapt to it and on later stage he can affect the team members that has not become found of the method.

5.5 Team Orientation

The final area in my framework discussed in chapter 3 is team orientation. Section 3.5 describes team orientation as:

"the ability to take other team member's behavior into account and set team goals over individual goals."

Team orientation is also well facilitated by Scrum. Scrum includes planning procedures and several meetings that facilitates team orientation. However, to achieve good team orientation Scrum leaves much influence in the hands of the team. Below I present and discuss my results concerning the last area from Salas' model.

The tables below present my findings related to team orientation:

Table 5.12: Literature Results1- Team Orientation

ID	Findings	Results
[38]	The study discusses developers working in a different role than usual in order to assist overloaded team members. From the study: *"... the team never worked overtime or weekends. Additionally, developers got to work outside their roles to help their Scrum team achieve its sprint goals."*	Productivity, morale and team effectiveness increased.
[35]	After an episode where the scrum master was not present, developers started using the Scrum daily meeting as it should be used. Together they found out that they wanted to work more as a unit and understand what the different developers are working on.	Internal communication improved and results improved with time!
[42]	Introduced "Project Concern" which is a functional aspect concidered to be of such importance that it should be treated seperately and be visible to everyone.	Defining such concerns strengthened team orientation
[42]	Introudced "The Wall" which categorized the remaining tasks into "not started", "in progress" and "finished". The wall was visible to the entire team.	The developers got a shared vision of what was left of work. Effectivity increased slightly.
[42]	An unrealistic plan made the team members prioritize individual goals over team goals.	Developers started choosing the most interesting tasks instead of the ones with the highest priority.

Table 5.13: Literature Result2 - Team Orientation

ID	Findings	Results
[42]	Team meetings where everyone participated were ineffective and without a clear agenda. The discussions concerned only the project owner, project leader and the Scrum Master. The topics often totally excluded the developers.	This affected the team's chance of really committing to the team goals and plans, which is necessary for achieving team orientation. Performance slowly decreased.
[3]	In this case, the team barely participated in decision making during planning meetings. The most experienced developer took all the decisions.	Scrum's daily stand-up meetings became uninteresting, the developers cared less for the project and productivity decreased.
[35]	Developers prioritized individual goals over team goals. A developer stated: *"People working alone results in the team not discovering problems, because you do not get feedback on your work."*	Lack of tem orientation weakened the team work, thus team effectiveness decreased.
[35]	In this study team members did not feel comfortable with all the other team members. The researchers found that the team was lacking trust and team orientation. One of the interviewed developers stated: *"There is no problem getting criticism from people you feel safe with, but when you get feedback from people you do not like, it is different"*	Feedback within team members were reduced. Quality of project was most likely reduced.

Table 5.14: Case Results - Team Orientation

ID	Collection Method	Findings	Results
#1	Observation and Interviews	Developers not working 100% on this case. Scrum master is working 60% and one of the developers IS working as little as 20% on the case.	This affects team goals not being prioritized because there are other projects that also require attention. This affects the wealth of team orientation.
#2	Observation from retro-spective and planning meetings	Some developers seemed uninterested in the discussions.	This affects the individual developer's chance of committing to the team goals. This weakens the team's chance of achieving good team orientation.
#3	Observation from planning meetings	During the planning poker process a developer from the client side was observed "cheating". The developer repeatedly looked over at the neighbor and then threw the same estimation card on the table as he did.	This also affects the individual developers chance of committing to the team goals - weakening the chance of good team orientation.

From what I have found out in literature and the analasis of my case the general feeling is that teams struggle to achieve good team orientation. Team orientation is facilitated in several areas of the different teams, but not on all areas. This chapter will take a further look into some of the different areas.

In [35] they discuss the lack of trust between the scrum master and the rest of the team members. The theory of teamwork from [1] states that trust is one of the core components in team work. This can also be seen from 3.1. It directly affects mutual performance monitoring, but affects all areas

within "the big five". Salas state that without sufficient amount of trust, team members will spend time and energy protecting their own work instead of collaborating to gain new ideas. The lack of trust was no issue in the studied case. The developers felt comfortable talking to each other and to the scrum master and it showed that trust was there at all times during the project. This facilitated team orientation and also increased the team effectiveness for the project.

Team orientation is about having the ability to take feedback from other members and then use this feedback for own decision-making and work. In scrum it is organized to facilitate team orientation by using daily stand-up meetings. In my case, daily standup meetings are practiced every working day except the days of retrospective and planning meetings. If there are development issues they are brought up on these meetings. The scrum master often knows the answer or at least knows someone that can provide the answer. These daily standups obviously increased team effectiveness. Results from my literature part[35, 4] also show that teams not taking these meetings seriously struggle in becoming well functioning self-managing teams. Literature also showed that things were better after team members understood that daily meetings could work as a helping tool, which again proved to be even more useful in the later stages of the projects. A quote from one of the developers working on a project from [35]:

"The good thing about Scrum is that Scrum reminds us to talk to each other about the project."

This shows that the workers believed in Scrum and liked the idea to talk to each other about the project, even though their scrum master used little out of the Scrum guidelines. The team members showed that they wanted to work in a team. This resulted in increased team effectiveness.

Salas' [1] describes good team orientation as individuals wanting to work in a team and to set team goals over individual goals. From my literature results [35] the different teams struggle with this. This is shown in my case as well. An explanation of this can be that some of the developers are not working 100% on this case. From interviews and observation, my general impression was that the team members was interested and tried to work as a team, some factors impeded the team effort. These factors were some organizational factors concerning the paralell ongoing projects, as well as the lack of involvement in discussions from some members on the clients side. It is difficult to come forth and play a big part of discussions the other members have more knowledge about, however it decreases the team orientation when

the same people always are discussing. The idea of the planning poker tool is that it should facilitate adaptability and team orientation. However this is not entirely the case in this project. Even though the planning poker tool includes everybody, some developers from the clients side seemed uninterested. Also, as can be seen in 5.14 I observed a team members "cheating" in the planning process [Result #3]. This affects the individual chance of commiting to the team goals. The reasons for such behavior can be so much, however, ultimately it will decrease team orientation.

It is also interesting to see that the developers working less on the project seem to be less interesting in it as well [Result #1 and #2]. Having developers working as little as 20% on the project does not help for the learning process or team orientation. The scrum master says in an interview:

"The learning process is slow for some developers. The reason for this can be many things, but we just have to deal with it. We are still on time and will finish within time"

From observation it seems like scrum master is right about this. Even though the team lacks team orientation I doubt that team effectiveness was decreased too much. Knowledge and experience seem to be just as good to increase effectiveness.

Chapter 6

Conclusion

In this study I wanted to answer the question:

"How does the Scrum methodology facilitate team effectiveness and what are the advantages and disadvantages?"

To answer this question, the core areas from Salas' model were in focus. Previous research papers have been investigated and compared, a case study has been observed and analyzed and the results from both have been discussed and compared. I have found that teams were struggling to adapt to a new working methodology. Scrum as a methodology is working very well, but only after the developers get used to it. My literature results show that scrum works well in some areas of all the studied projects, but very well on all areas in another project. My observed study also shows that Scrum is working very well at some areas, but that there are room for improvement in other areas.

Leadership

Through this study the author has found that having an experienced scrum master is alpha and omega to success. My case study shows that the scrum master is one of their keys for a successfull project. Teamwork is easier when the developers know what should be done and when each task that should be completed are explained properly. The retrospective and planning meetings provided the team with a shared understanding for the upcomming sprint. Such shared understanding is explained in theory as a "shared mental model" that increases good teamwork. This is also the case for my observed team.

From my results in chapter 5.1 I can conclude that the self-managing team with the solid scrum master facilitate good leadership and that team effectiveness is increased. The results from my literature shows that having a scrum master not knowing how to use the scrum guidelines impacts the team to such an extent that team effectiveness greatly decreases. If the team is inexperienced and does not have any obvious choice for a scrum master a coach can guide the team to success. Having a coach to facilitate and help out, at least in the early stage of a project can be the difference between success and failure.

Mutual Performance Monitoring

One of my main findings for this area is that working in a closed room facilitates teamwork and increase team effectiveness. The easy access to help from other developers is very useful. In addition, I have found that the use of burn-down charts is a great tool to facilitate mutual performance monitoring. These helped the team knowing how much work that was left in each sprint and make the planning easier in the process. Overall I can conclude with the observed team had areas where mutual performance monitoring helped teamwork and efficiency.

Backup Behavior

In the literature I found that not having backup behavior resulted in tasks not completed and work being moved to the next sprint. My main findings is that having a plan for what happens when backup is needed is crucial for a project to be successful. The observed team helped each other out and the leader arranged for quick external support. I can conclude with the team having good backup behavior and that team effectiveness was increased because of this.

Adaptability

Adaptability was a challenging area in both the studied literature and in my observed case. My main finding is that in order to adapt properly to agile methods, interest and time is of big importance. In addition I found

that there are several ways to support adaptability. For example can pair programming be used as a tool to support the learning process and increase good teamwork. I also found that adapting to the different issues that might occur during a development project is very important for team effectiveness.

Team Orientation

Teams struggle with achieving good team orientation. One result is that developers struggle to set team goals over individual goals. This can be seen both in my observed case and in the studied literature. I also found that developers working less than 100% on a project can be an issue concerning good teamwork. For instance, there was lack of involvement in discussions. Theory from chapter 3.5 states that this would decrease team effectiveness. In my case I would argue that there was little choice of changing the way things were handled in order to increase effectiveness. I believe including everyone at all times, for my observed case, would cost the team more time and effort than necessary. When developers are working less than 100%, it comes a point were it is better for the leader to lead the discussions and push forward the important points, rather than having everybody to say their meanings. For my observed case I would say that Salas' teamwork model would not be the best choice.

Scrum and Teamwork

Overall, I have found that Scrum facilitates teamwork and will in most areas increase team effectiveness.

For example, Scrum facilitates involvement for discussions. Having negotiating skills also proves to be very important in teamwork. Everybody in the team should be forced to join discussions when decisions should be made. Using the scrum daily meetings can be a good way of improving these skills. Joining discussions also facilitates several areas of Salas' model.

My studies show that it can be hard to adapt in to agile development methodologies. However, the studies also show that when using agile development, such as Scrum, and follow the guidelines correctly, teams are satisfied and productivity improves. Concidering this, there are still things that can be improved and several areas around agile development that can

be discussed and be made further research on.

Salas' Model

Salas et al. 2005 and their statements concerning teamwork are correct according to what I have found out. However, according to team effectiveness I can argue that not all areas are answered. Within my literature studies when there is a lack of team orientation, good leadership, lack of backup behavior, mutual performance monitoring or adaptability within the team, team effectiveness is decreased. Concidering my studied project it is not allways like this. For instance there were issues concerning team orientation and mutual performance monitoring where Salas' would argue that team effectiveness would decrease. In my observed case study, I could argue that Salas' model might not be the best model to measure effectiveness for this specific project.

Chapter 7

Future Work

The intention of my work was to study several literary papers, investigate a software development project and compare these to draw conclusions concerning the areas of Salas' model. As my findings from this report are based on quite few studies and only one case study, I suggest adding more cases as a basis for the analysis. I would also recommend using another model in addition to Salas' as a framework for project teamwork theories to analyze more concerning team effectiveness.

Bibliography

[1] Eduardo Salas, Dana E. Sims and C. Shawn Burke. Is there a "big five" in teamwork? 2005. Online version can be found at URL: *http://sgr.sagepub.com/content/36/5/555*

[2] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. Information and Software Technology, SINTEF ICT, S.P. Andersensv. 15B, NO-7465 Trondheim, Norway, 2008.

[3] Nils Brede Moe and Torgeir Dingsøyr, Scrum and Team Effectiveness: Theory and Practice, NO-7465 Trondheim, Norway, 2008.

[4] Nils Brede Moe, Torgeir Dingsøyr, Tore Dybå, Understanding Self-organizing Teams in Agile Software Development, IEEE Computer Society Washington, DC, USA ©2008

[5] David Cohen, Mikael Lindvall and Patricia Costa. An introduction to agile methods. Fraunhofer Center for Experimental Software Engineering 4321 Hartwick rd, Suite 500 College Park, MD 20742 USA, 2004

[6] G. L. Stewart and C. C. Manz. Leadership for self-managing work teams: A typology and integrative model. 1995

[7] S. J. Zaccaro, A. L. Rittman and M. A. Marks. Team leadership. Leadership Quarterly. 12, 451–483, 2001

[8] Cathy C Durham, Don Knight, Edwin A Locke. Effects of Leader Role, Team-Set Goal Difficulty, Efficacy and Tactics on Team Effectiveness. 1997

[9] Kevin Vlaanderen, Slinger Jansen, Sjaak Brinkkemper and Erik Jaspers, The agile requirements refinery: Applying SCRUM principles to software product management, *Department of Information and Computer SciencesUtrecht University*, 2010

[10] Yukl, G. A. Leadership in organizations. Englewood Cliffs, NJ: Prentice Hall. 1989

[11] R. M . McIntyre and Eduardo Salas. Measuring and managing for team performance. ISBN: 978-0-76231-141-5, 1995

[12] Cooper, R., & Sawaf, A. Executive EQ: Emotional intelligence in leadership and organizations. New York: Grosset/Putnam. 1996

[13] Robert R. McCrae & Oliver P. John. An Introduction to the Five-Factor Model and Its Applications. 1992

[14] Susan L. Kichuk & Willi H. Wiesner. The Big Five personality factors and team performance: implications for selecting successful product design teams. 1997

[15] Judy Kay, Nicolas Maisonneuve, Kalina Yacef, Peter Reimann. The Big Five and Visualisations of Team Work Activity. 2006

[16] George A. Neuman, Stephen H. Wagner and Neil D. Christiansen. The Relationship between Work-Team Personality Composition and the Job Performance of Teams. 1999

[17] Arthur E. Poropat. A Meta-Analysis of the Five-Factor Model of Personality and Academic Performance. 2009

[18] H. van Vliet, A decade of agile methodologies: Towards explaining agile software development. journal homepage: www.elsevier.com/locate/jss. 85 (2012) 1213– 1221

[19] Webber, S. S. Leadership and trust facilitating cross-functional team success. Journal of Management Development.Vol. 21 Iss: 3, pp.201 - 214, 2002

[20] Bandow, D. Time to create sound teamwork. The Journal for Quality and Participation. 24(2), 41-47, 2001

[21] S. J. Zaccaro, A. L. Rittman and M. A. Marks. Team leadership. Leadership Quarterly. 2001

[22] M. A. Campion, G. Medsker and C. Higgs. Relations between work group characteristics and effectiveness: Implications for designing effective work groups. Personnel Psychology. 1993

[23] J. E. Driskell and Eduardo Salas. Collective behavior and team performance. Human Factors,Volume: 34, Issue: 3, Pages: 277-288, 1992.

[24] Eby, L. T., & Dobbins, G. H. Collectivistic orientation in teams: An individual and group-level analysis. Journal of Organizational Behavior, 18, 275-295, 1997

[25] Randy J. Larsen & David M. Buss. Personality Psychology - Domains of Knowleedge About Human Nature. Fourth Edition. Chapter Three (New York, Published by McGraw - Hill). 2010

[26] John, O.P. The "Big Five" factor taxonomy: Dimensions of personality in the natural laguage and questionnaires. In L.A. Pervin (Ed.), Handbook of personality (pp. 66-100). New York: Guilford Press. 1990

[27] Saucier, G., and Goldberg, L.R. What is beyond the Big Five? Journal of Personality, 66, 495-524. 1998

[28] Wiggins, J. S. The five-factor model of personality: Theoretical perspectives. New York: Guilford Press. 1996

[29] Block, J. Going beyond the five factors given: Rejoinder to Costa and McCrae (1995). Psychological Bulletin, 117, 226-229. 1995

[30] McAdams, D. P. The five-factor model in personality: A critical appraisal. Journal of Personality, 60, 329-361. 1992

[31] Bob Schatz and Ibrahim Abdelshafi, Primavera Gets Agile: A Successful Transition to Agile Development, May/June 2005 (vol. 22 no. 3).

[32] Brian Fitzgerald, Gerard Hartnett and Kieran Conboy, Customising agile methods to software practices at Intel Shannon, Volume 15 Issue 2, April 2006 Pages 200 - 213.

[33] Nina Dzamashvili Fogelstrom, Tony Gorschek, Mikael Svahnberg and Peo Olsson,The impact of agile principles on market-driven software product development, DOI: 10.1002/spip.420, 2009

[34] Nils Brede Moe, Torgeir Dingsøyr and Tore Dybå, A teamwork model for understanding an agile team: A case study of a Scrum project, SINTEF, NO-7465 Trondheim, Norway, 2010

[35] Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå, Overcoming Barriers to Self-Management in Software Teams. Volume: 26, Issue: 6 Page(s): 20 - 26, 2009

[36] Dingsoyr Torgeir, Nerur Sridhar, Balijepally VenuGopal, A decade of agile methodologies: Towards explaining agile software development, SINTEF, NO-7465 Trondheim, Norway, 2012

[37] Michael D. Myers, Investigating information systems with ethnographic research.Volume 2 Issue 4es, Dec. 1999 Article No. 1

[38] Frances Julia Riemer, Ethnography Research. Ph.D. 6. Riemer, F, 2008

[38] Bob Shatz and Ibrahim Abdelshafi. Primavera Gets Agile: A Successful Transition to Agile Development, IEEE SOFTWARE Published by the IEEE Computer Society. 0740- 7459/05/20.00 ©2005

[39] Nina Dzamashvili Fogelstrom, Tony Gorschek, Mikael Svahnberg and Peo Olsson., The impact of agile principles on market-driven software product development, DOI: 10.1002/spip.420, 209

[40] Brian Fitzgerald, Gerard Hartnett and Kieran Conboy. Customising agile methods to software practices at Intel Shannon. Volume 15 Issue 2, April 2006 Pages 200 - 213.

[41] Artem Marchenko and Pekka Abrahamsson, Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges. ISBN: 978-0-7695-3321-6, 2008

[42] Viktoria Gulliksen Stray, Nils Brede Moe, and Torgeir Dingsøyr. Challenges to Teamwork: A Multiple Case Study of Two Agile Teams. Pages: 146-161, 2011

[43] Torgeir Dingsøyr and Tore Dybå. Team Effectiveness in Software Development - Human and Cooperative Aspects in Team Effectiveness Models and Priorities for Future Studies. NO-7465 Trondheim, Norway.

[44] Jeff Patton, Agile Product Design. 2009, http://www.agileproductdesign.com/blog/2009/kanban_over_simplified.html

[45] James J. Jiang, Jaideep Motwani and Stephen T. Margulis (1997). IS team projects: IS professionals rate six criteria for assessing effectiveness Team Performance Management

Chapter 8

Appendix A - eXtreme Programming

Extreme Programming is the most researched field within agile methods [2]. It provides a clear description on how projects should be run and is based on existing methodologies. This is shown in 8.1 below:

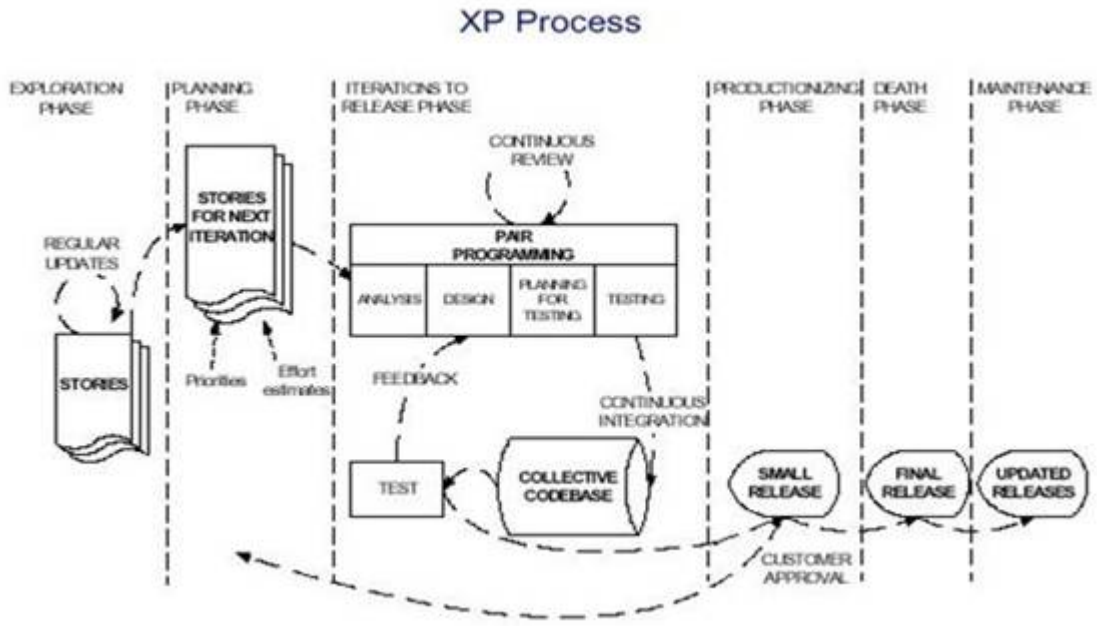

Figure 8.1: The life cycle of XP

The XP methodology is mainly for small to medium sized teams which work with technology that allows changes when the customer asks for a change in the requirements. The guidelines for XP are important in several areas of a project. I will not go into all of them in this report, but I have listed the most essential below:

1. ***The Planning game:*** Developers and customers cooperate to decide scope and timing and estimate the amount of work.

2. ***Pair programming:*** Two people produce code from one work station

3. ***Continuous releases:*** New features or versions of a system are released and can be used right away by the customer

4. ***40-hours week:*** No team member can work more than 40 hours each week. Anything more than this will be seen as a problem for the entire team.

5. ***Shared Workspace:*** The team works together in an open workspace which makes it easier to cooperate. This also makes it possible for any team member to change any code within the project at any time.

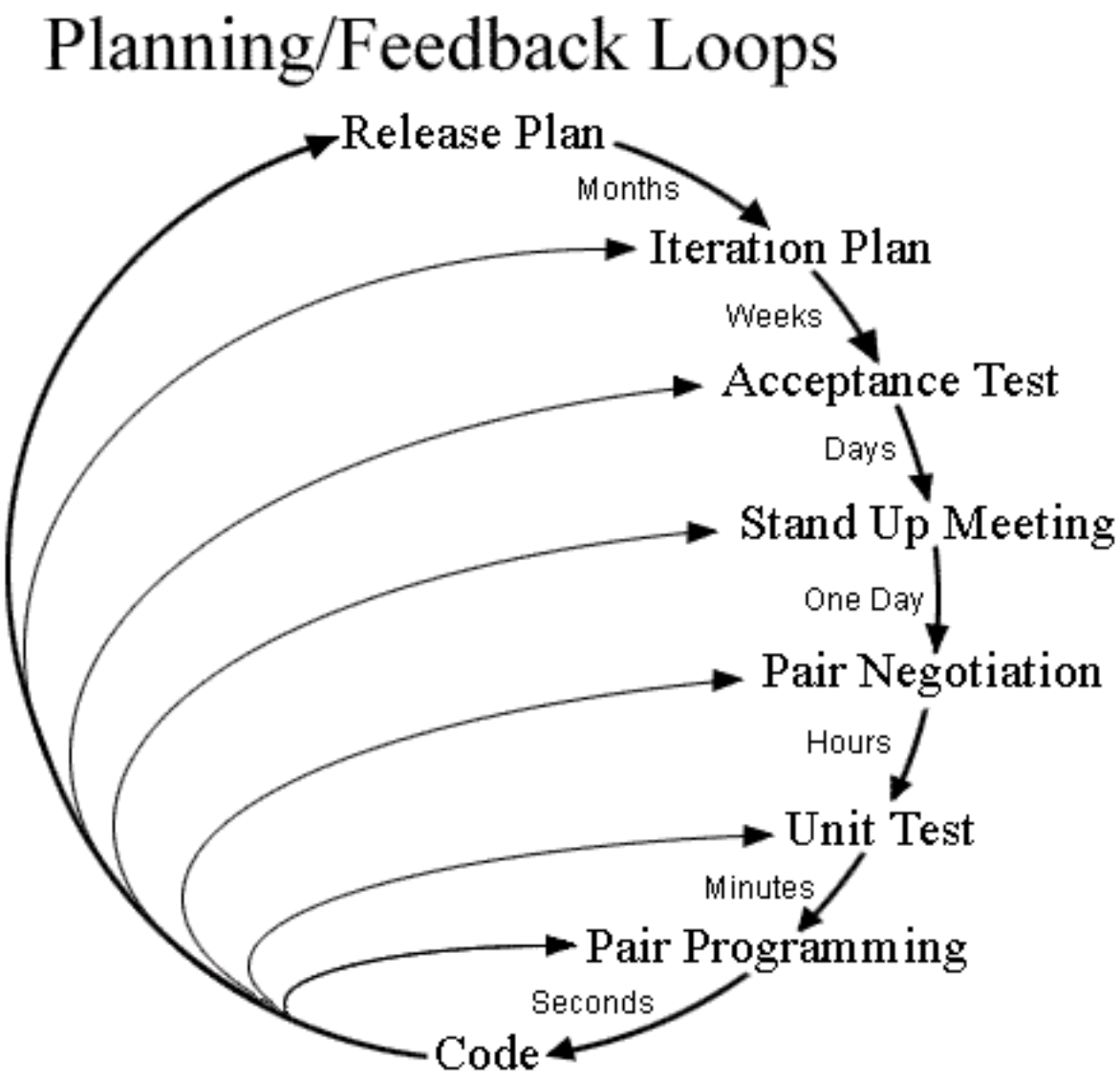

Figure 8.2: Planning/Feedback Loops in XP

Chapter 9

Appendix B - Interview Guides

In this appendix I will present the different interview guides that were used. All the interviews were done in norwegian.

Interview Guide for Project Leader - Client

This is an interview guide for the interview with the project leader - client side. The interview itself is much longer including a lot more questions. In this interview I wanted to focus on getting a good overview of the project and the project leader's work and impact on the project.

1. Hva går prosjektet ut på?
2. Hvor mye jobber du på prosjektet?
3. Hva er dine oppgaver? Hvordan utfører du dem?
   (a) Oppfølging: Er du noe med utvikler teamet og når eventuelt?
4. Hva slags erfaring har du med utvikling fra tidligere?
5. Kan du beskrive hvordan kommunikasjon mellom deg, teamet og prosjektleder til Capgemini foregått?
   (a) Oppfølging: Gi eksemple
   (b) Oppfølging: Beskriv hvordan man løser problemer når de oppstår
   (c) Oppfølging: Få en hierarkisk oversikt til prosjetet

6. Hvordan synes du kvaliteten er på det som er utviklet har vært til nå?
7. Hvordan synes du effektiviteten er?
8. Hva har fungert særdeles bra i prosjektet?
    (a) Gi eksempler!
9. Hva har fungert mindre bra?
    (a) Gi eksempler på problemer som har oppstått
10. Etter prosjektet er ferdig. Hva skjer videre?
11. Hvordan har dette prosjektet vært sammenlignet med andre prosjekter?

Interview Guide 1 for Project Leader - CC

Interviewguide for the project leader from the consultant company. This interview also focus on getting an overview of the project - from the consultant company's perspective - and the project leader's impact on the project. It is interesting to see whether there are any differences among the project leaders context of the project.

1. Hvor lenge har prosjektet foregått?
2. Når skal prosjektet avsluttes?
3. Hvordan kom dere i kontakt med kunden?
4. Hvorfor valgte dere å gjennomføre prosjektet?
5. Hvordan foregikk utforming av kravspesifikasjon og planleggingsfasen?
    (a) Gi eksempler
6. Hvordan har oppgaver blitt oppdelt og tildelt?
    (a) Gi eksempler
7. Hvordan har dette prosjektet vært sammenlignet med andre prosjekter?

8. Hvor viktig har prosjektet vært for firmaet?
    (a) Oppfølging: Vil finne ut om firmaet prioriterer dette prosjektet høyt?
9. Hvordan synes du kvaliteten er på det som er utviklet?
    (a) Oppfølging: Hva tror du synes kunden om kvaliteten?
10. Hva har fungert særdeles bra i prosjektet?
11. Hva har fungert mindre bra?
12. Hvor ofte har du kontakt med kunden?
13. Har det vært endringer av kravspesifikasjon?
    (a) Oppfølging: Hva slags endringer.. Hvordan løser dere dette?

Interview Guide Scrum Master

Interview guide for an interview with the Scrum Master. This interview focuses on how the work is distributed and performed during the project. In addition it also focuses on Scrum Master and what experience he has within agile development.

1. Jobber du bare på dette prosjektet?
    (a) Følg opp: Hvor mye evnt av tiden hans til dette og hvor mye til andre. Mye overtid?
2. Hva med resten av teamet?
    (a) Få med hvem som jobber og hvor mye
3. Hvordan har oppgaver blitt oppdelt og tildelt?
4. Hva slags erfaring har du med smidig utvikling?
    (a) Scrum?

5. Følger teamet Scrum guidelines?
    (a) Hvordan?/Hvordan ikke? Eksemplifiser!
6. Hvordan føler du samarbeidet innenfor teamet fungerer?
7. Følger dere de daglige møtene?
    (a) Ja/nei?
    (b) Hvordan fungerer dette?/Hvorfor ikke?
8. Hvordan ble estimeringsprosessen gjennomført?
    (a) Hva synes du om denne? Få positive og negative sider hvis de finnes!
9. Fungerer bruken av backloggene?
    (a) Hvordan ble de planlagt?
    (b) Hvordan blir det brukt?
10. Hvor viktig har det vært å fullføre sprint backloggene?
11. Blir burndown charts brukt i prosjektet?
    (a) Hvis – hvordan? Synes du disse er nyttige?
12. Hvordan har dette prosjektet vært sammenlignet med andre prosjekter?
13. Hvordan synes du kvaliteten på det som er utviklet er?
14. Hva synes teamet om kvaliteten?
15. Hva har fungert særdeles bra i prosjektet?
    (a) Eksempler!
16. Hva har fungert mindre bra?
    (a) Eksempler!
17. Har det vært mye endringer i kravspesifikasjonen?
    (a) Hvordan løser dere dette i såfall?
    (b) Påvirker dette prosjektet i stor grad? Forklar!

Interview Guide 2 - CC

Interview guide number two for the project leader and Scrum Master from the consultant company. This interview focuses on teamwork and Scrum, having Salas' model [1] in focus. It is interesting to see whether the Scrum Master and project leader have the same opinions about the development team.

Leadership:

- Hvordan synes du prosjektet har gått?
  - Positive/Negativer sider. Mulighet.
  - Effektivitet?
- Har lederen fasilitert og koordinert prosessene på en positiv måte?
  - Eksemplifiser!
  - Hva har vært bra her?
  - Mindre bra?
  - Felles modell?
- Har teamet hatt en felles forståelse av prosjektets fremgang?
- Hvordan har beslutninger blitt tatt?
- Tror du teammedlemmene føler de har bidratt til å styrke prosjektet?
- SSO – henter dere dere inn igjen?

Mutual Performance Monitoring:

- Tror du folk i teamet vet hva de andre i teamet holder på med av oppgaver?
  - Hvis ikke detaljer nok så er det helt fair
- Føler du at folk i teamet har kunnet stole på hverandre?

Backup Behavior:

- Det var en periode dere trengte mer arbeidskapasitet – hvordan taklet dere dette?
- Hvordan ble dette tatt i mot av teamet?

Adaptability:

- Hvordan har de uerfarne utviklerne (ift Scrum) tilpasset seg Scrum?
  - Har dette gått fint? Eksempler!
- Har det vært andre måter teamet har måtte tilpasse seg ting på?

Team Orientation:

- Har andre oppgaver påvirket dette prosjektet på noen måte?
  - andre prosjekter osv..?
- Hvordan inntrykk har du av teamet fra standup-møtene?
  - Kan også ta opp demo-møter, retrospektive, planning meetings og sammenligne!
- Hvordan føler du kompetansen i teamet har vært distribuert?
- Føler du dere har levert det dere har tenkt?
- Hvordan tenker du fordelingen av arbeidet til kunden sine konsulenter har vært?
- Tror du en annen måte å fordele resurrsene kunne vært en ide?
  - F.eks la de bare jobbe sprint 1-5..? Hvorfor/Hvorfor ikke?

Til Prosjektleder anngående oppgaven min:
Trenger jeg mer til contexten anngående releases til sluttbrukerne?
Beskriver contexten prosjektet på en oversiktlig måte?