NTNU

Norwegian University of
Science and Technology

Software Contracting and Agile Development in the Norwegian ICT Industry

A Qualitative Survey

Anders Ganes
Snorre Nævdal

Master of Science in Computer Science

Submission date:	June 2008
Supervisor:	Reidar Conradi, IDI
Co-supervisor:	Hanssen Geir Kjetil, IDI

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

Problem Description

As more and more ICT companies migrate to agile methods, the existing contracts and contractual frameworks are mostly tailored for plan-based development. It has been indications from the industry that there are several challenges using agile methods with various contract standards available today. This assignment aims to find out what the potential challenges are with using today's contract models with agile development. In addition, the assignment will look into what work is done in adapting contract standards to better comply with agile principles.

Assignment given: 16. January 2008
Supervisor: Reidar Conradi, IDI

Abstract

This thesis takes a closer look at how various contract models affect the ability to use agile software development, and the work being done to help improve contracts so they better facilitate agile principles.

Agile development methods are becoming increasingly popular, while most contractual standards are meant for plan-based development methods. There are done little research regarding the subject, but it is a growing concern in the industry. Work is also currently done on Norwegian contractual frameworks that aim to update two commonly used contracts.

The research done consist of several interviews with industry practitioners representing Norwegian consultancy companies experienced in agile development methods. It also includes interviews of contract experts in addition to a search and study of the literature on the subject. These interviews have been analyzed and compared to literature on the subject.

The results of the research revealed a number of challenges with today's contract models. Several discoveries showed difficulties with bureaucracy and customer involvement. Using traditional contracts could also easily require more upfront work than what is sensible when using agile methods. It is possible to bypass these challenges by either adapting the development method, or putting the contract aside.

Preface

The assignment for this thesis is given by Department of Computer and Information Science (IDI) at the The Norwegian University of Science and Technology (NTNU). The context of the work is within the EVidence based Improvment of SOFTware engineering (EVISOFT) project, partly funded by the Research Council of Norway (Norges Forskningsråd) under Grant 174390/140. The project has as its goal to improve software engineering processes, and to do this to such a degree that it becomes economically beneficial.

The work on this thesis has progressed through the spring of 2008 and is part of the our MSc in Computer Science. Our work consists of a literature study, and a survey of both the experience with different contracts for software development in combination with agile development methods, and the work done to make two Norwegian contractual standards more compliant to agile principles.

The thesis started as a case study at a company called Ciber, who approached the institute with several research topics they would be interested in pursuing along with students at NTNU. After a review of the topics the authors along with the co-supervisor Geir Kjetil Hanssen, agreed to focus the thesis on the potential challenges associated with traditional contracts used in agile projects. The work was planned as a case study using several available projects at Ciber, but after the first round of interviews it became clear that the participants with most experience on the subject were people involved in sale, and negotiation of contracts. We therefore expanded the interviews to include people with this experience in other companies. During the literature study we also discovered the currently ongoing work that Mari Vestre and PROMIS were doing to facilitate two Norwegian contractual standards, respectively The Norwegian Governmental Standard Agreement for System Development (agile version) (SSA-S) and Project Management 2000 (PS2000) for easier use with agile methods. This discovery lead us to focus on this aspect as well.

We would like to thank our supervisor Reidar Conradi. We would also like to thank Ciber, represented by Marianne Selle and Sigurd Gimre, for suggesting

both the cooperation we enjoyed and the basis for the research. Thank you to all the participating companies and interviewees, whose experiences and input on the subject has been invaluable. In particular the people who welcomed us warmly at Ciber (Marianne Selle), ObjectNet (Sverre Tinnen and Lars Ivar Næss), Bekk (Reidar Sande), Promis (Kjetil Strand and Jørgen Petersen) and DIFI (Mari Vestre).

Finally, we would like to thank our co-supervisor Geir Kjetil Hanssen for his guidance and valuable insights on the subject. He has been a driving force throughout the entire process of writing this thesis, and his comments and feedback have been invaluable to us.

Trondheim, June 2008

Anders Ganes   Snorre Nævdal

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The technological development is gaining speed while more and more of its momentum relies on software. According to Robert Charette, a typical cellphone made in 2005 contains 2 million lines of code, while an ordinary car might contain as much as 100 million lines of code. This explains why we have come to a point where the average company spends approximately 5 percent of their revenue on Information Technology (IT). The highly IT dependent companies can use as much as 10 percent, making it one of the largest corporate expenses outside employee costs [6].

Another study by Kjetil Moløkken-Østvold et.al summarize the estimation techniques used in 26 Norwegian companies, and discovers that 62% of the projects were plagued with missed deadlines. This is in accordance to similar previous studies mentioned. In order to improve the work done within the field of software engiennering, a myriad of development tools, techniques, methods and processes has over the years been proposed and created. One of the recent proposals for improvement has been labled agile development methods and have created a buzz in the software engieneering community. Many of these agile methods are the result of experiences made by respected practitioners who have honed their development methods through successes and failures.

The simplicity and practical approach of these methods has made them increasingly popular and they have by now made a huge impact on the development of software world wide. Having become a viable choice for most projects, more and more are questioning the claimed rewards of using agile practices. This has caused a widespread discussion of how one best can take advantage of agile development and when it is sensible to use these methods.

1.2 Project context

This report is part of the EVISOFT project. It is a joint project between the following research institutions: SINTEF ICT, NTNU, SIMULA research center and The University of Oslo (UiO).

It is partly financed by the Research Council of Norway (Norges Forskningsråd) under Grant 174390/140 and currently sponsored by the following Norwegian Information and Communication Technology (ICT) companies: ABB, DNV Software, EDB, CONFIRMIT, Geomatikk, Konsberg Spacetec, KnowIT-Objectnet, Software Innovation, Telenor and Vital. These sponsors all contribute by committing themselves to spend time on testing practices recommended or supported by the EVISOFT project.

The EVISOFT project is the successor of the project SPIKE (2003-2006) which in turn was the successor of PROFIT (2000-2002). All these projects share a common goal, namely to improve processes in software-intensive companies. The project is going to achieve this by monitoring and collecting evidence from the contributing ICT companies while they use state-of-the-art tools, methods or processes. While monitoring these companies the project will gain knowledge that after analysis might reveal strategies that can ensure future successes or predict possible failures.

1.3 Problem definition

It was decided to focus the research on the management of agile projects, with main focus on the impact contract models has on agile methods. In such a study, it is difficult to know or predict what results you are going to get, and it is therefore difficult to define a very specific and concrete research question. It is likely that the research uncover various interesting aspects with agile development, and it is important that all of these aspects are included in the report, and not omitted because they are irrelevant because of a too specific research question.

As more and more ICT-companies migrate to agile methods, the existing contracts and contractual frameworks are mostly tailored for plan-based development methods. This research will try to discover how the Norwegian industry experience this, and what potential solutions exist.

There are several reasons why the research is aimed at agile methods and contract models. Firstly, there has been indications from the industry that this is a very interesting and relevant topic. Secondly, there exist little research on the topic today.

The following research questions are defined:

1. What are the experiences with using agile methods and how does the contractual model used in a project affect its ability to use such methods?

2. What work is done in adapting contract models and standards to agile methods in the Norwegian ICT-industry?

1.4 Scope

The scope of this thesis will include software development concultancy companies who uses software development contracts in their cooperation with customers. These software suppliers all have considerable experience with agile development methods.

This thesis does not include customer experiences. Due to the timespan of the thesis, details regarding the supplier-customer relationship and juridical aspects of the different contracts have been left out.

1.5 Document Outline

The outline for the rest of the report is as follows:

- Chapter 2: Background
  This chapter takes a look at agile development, and how is has evolved. It also gives a quick introduction on how it works, what research exist regarding the subject, and the motivations and challenges users of the methods face.
- Chapter 3: State of the art
  This chapter summarizes the research literature found on agile development and contract models.

- Chapter 4: Research
  This chapter contains an overview of the research done in this thesis. It is divided into six parts covering the the description of the goal of the research and the research questions. It continues to review the considerations made when choosing research approach, and how the research is designed. It also describes how the research process proceeded, and lastly this thesis' limitations and biases.
- Chapter 5: Results
  This chapter presents the results from the interviews. The results are organized by interview subjects, and contains a summary of the interview to improve readability.
- Chapter 6: Analysis
  This chapter organize the results by different identified topics in such a way that all statements concerning a topic is assembled.
- Chapter 7: Discussion
  This chapter discusses the analysis and compares the results from findings in the literature.
- Chapter 8: Conclusion
  This chapter presents the final conclusions of the thesis, and make suggestions for further research.

Chapter 2

Background

The rise of agile methodologies is often seen as a reaction to the most common approach of developing software, often called the traditional development methods or plan based development methods. For many practitioners the appeal of these agile methodologies is their reaction to the bureaucracy of the engineering methodologies. These new methods attempt a useful compromise between no process and too much process, providing just enough process to gain a reasonable payoff.

The agile methods are not entirely new however. The biggest source of ideas behind them lies in previously faulty models. Popular processes reveal their successes and failures through research or experience, and thereby help the software development process improve. What makes agile development stand out from other evolved models are the core values and principles that bind these methods together and the way they appeal to developers everywhere. In February 2001 a group of contributors within the field of alternative development methods gathered and created "The Agile Alliance". During their first meeting they also wrote the "Agile Manifesto" which consists of a four item list of general statements and 16 more detailed principles that represent the idea behind the term "agile".

2.1 Agile Manifesto

In February 2001 a group of software development practitioners gathered to discuss alternative development methods and try to find common ground. These were representatives from several of the well-known methodologies and others sympathetic to the idea, namely unite the alternatives to plan based development methods. This gathering led to the realization that even though many of the participants were competitors daily, they had compatible values and principles. Jim Highsmith, one of the participants, writes in his summary that the participants all agreed that a development environment is at its best when it *"does more than talk about 'people as our most important asset' but actually 'acts' as if people were the most important, and lose the word 'asset'"* [11]. The end result of their gathering is a manifesto consisting of four values and twelve principles which form the foundation of the agile movement. [1]. The signatories are representatives from eXtreme Programming (XP), SCRUM, Dynamic Systems Development Method (DSDM), Adaptive Software Development, Crystal, Feature Driven Development (FDD) and Pragmatic Programming.

The four values are quoted below:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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*

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	*Jim Highsmith*	*Steve Mellor*
Mike Beedle	*Andrew Hunt*	*Ken Schwaber*
Arie van Bennekum	*Ron Jeffries*	*Jeff Sutherland*
Alistair Cockburn	*Jon Kern*	*Dave Thomas*
Ward Cunningham	*Brian Marick*	*James Grenning*
Martin Fowler	*Robert C. Martin*	

Item three in this list states that customer collaboration is valued more than contract negotiation. This statement is part of the basis for this thesis as it points out one essential difference between traditional and agile development. The idea of involving the customer to a larger degree is essential, and logical. There are nobody with more knowledge about the wanted product than the customer. They might not have it exactly specified, or not even get it right the first time,

[1]`http://agilemanifesto.org/principles.html`

and they most probably will change their minds. This is why communication is essential throughout the process, and why even though defining everyone's rights and responsibilities through a contract is important, it is not a good substitute for communication.

The twelve principles in the agile manifesto:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity – the art of maximizing the amount of work not done–is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

2.2 Agile terminology

Among the most well known of the agile methods are eXtreme Programming, Dynamic Systems Development Method, SCRUM, Feature Driven Development, Crystal Methodologies and Lean software development. These methods are all categorized as evolutionary, iterative or incremental development methods. These three terms have very similar characteristics and are often used inconsistently. Incremental and iterative development are especially used interchangeably, they are not, however, synonyms. Iteration refers to the cyclic nature of a process in which activities are repeated in a structured manner. An increment refers to the quantifiable outcome of each iteration. The term evolutionary on the other hand has the following definition: "the development of usable software sub-systems in very small steps, with user requirements being continually re-assessed after each delivery" [13] and can be viewed as an collective term including all three.

2.3 Development process

Agile development can be described as the development of software in small iterations lasting one to four weeks where each iteration results in usable software. Each iteration is a development cycle in itself and can include planning, requirement analysis, design, coding and documentation. After each of these iterations the goal is to have an available release without bugs which the developers can present to the stakeholders in the project. This is usually followed by a meeting where developers asses their performances, and where a re-evaluation of the next iteration's priorities is done. A schematic illustration of a typical agile life cycle is presented in Figure 2.1 and is described in further detail in Section 2.3.1.

Agile development methods value face-to-face communication which means that sharing of information is done rapidly, but this form of communication does also assume that all of the participants are co-located. Another principle that affects the process is the frequent delivery of working software. Combined with face-to-face communication this ensures that the process produces far less written documentation compared to other development methods.

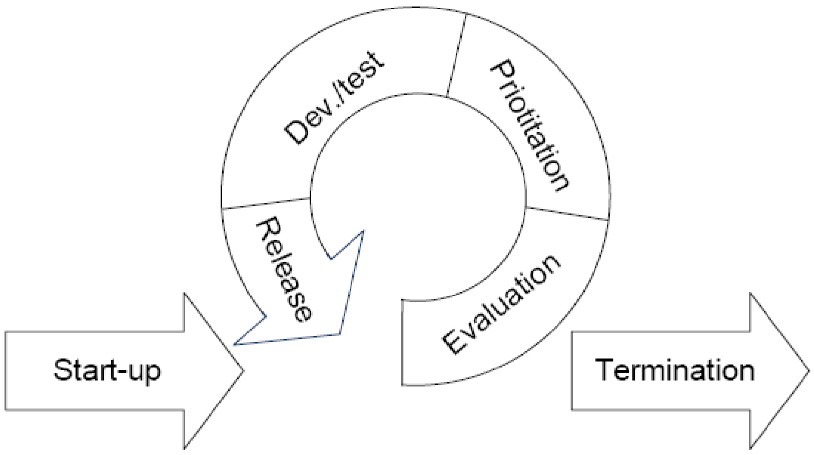

Figure 2.1: The agile lifecycle

2.3.1 Start-up

During the start-up one tries to get the initial funding for the project. Doing this will probably require an estimate as to how much the project is going to cost and how long it is going to last which can be difficult to answer since there are bound to be changes, and little is known. Customers open to agile methods do however have to settle for an evolving answer. The next step is to make an initial scope of the system in close collaboration with the stakeholders. This will result in a list of high priority items or use-cases that needs to be included and a rough sketch of the system being developed. When this is done an initial model of the architecture is created and the team, and resources needed is put together.

2.3.2 Evaluation

After each iteration all participants are invited to evaluate the progress and the experience this far. The evaluation is often informal and is used to present what has been accomplished in the current iteration by demonstrating new features or simply present the product as it is. This is usually followed by a review from all the team members where the process is evaluated. This can result in measures which the team wants to start doing or continue to do, or it can pinpoint issues that needs dealing with. This helps the team improve both the experience, and possibly the efficiency going into the next sprint.

2.3.3 Prioritisation

As a new iteration starts a collaborative effort is done to create the list of items that will be included in the next sprint. Much of the prioritisation is done by the customer who chooses the items that are most essential to the system. The software developers then break these into smaller packages which they then estimate the workload for. The limit of packages is determined by the amount of work hours available and can therefore add restrictions the customer must take into account. After the list of items is decided upon the team can analyze the items and make preparations such as design and modeling of the upcoming iteration.

2.3.4 Development / testing

The development of the system proceeds and the items within the iteration is implemented. During the implementation several techniques are suggested from multiple development methods. The most well known are however taken from XP and include pair-programming, frequent refactoring, collective code-ownership and continuous integration. The features that are added is usually well tested, and documentation is often written simultaneously. While the development cycle progresses, quick daily meetings are common. These meetings serve to report the progress to the participants and help reveal potential obstacles.

2.3.5 Release

Due to continuous testing during development the release is often done quickly. Acceptance testing is often done at this point to check whether all the features that have been implemented have been done so according to specification. Depending on the customer the release is delivered or presented, and a new evaluation begins.

2.3.6 Termination

When the system is completed the final testing commences. Potential errors discovered at this time are corrected and the documentation is finalized. It is also common to review the entire process in order to adjust behavior in later projects.

2.4 Agile in research literature

Agile development methods have become increasingly popular as a research subject. Searching the term *'agile development'* yield 64 500 hits[2], a number that seems to be steadily increasing. This trend is confirmed when looking at articles published and made available to Google Scholar the last 13 years, a search for articles published in each of the years following 1995 yields the results found in Figure 2.2.

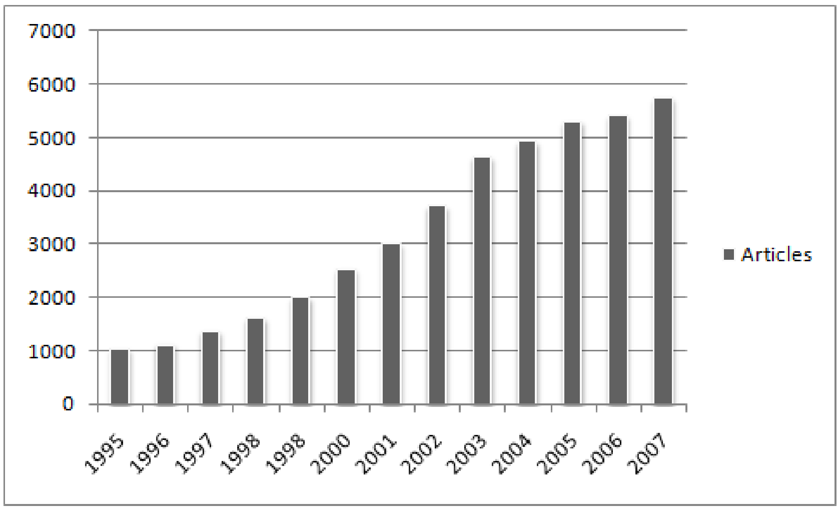

Figure 2.2: Articles regarding agile development

The use of agile development methods has also become increasingly popular in the industry. A study done by Scott Ambler in August 2006 shows that of their 4232 respondents there were 65% which had adopted one or more agile development techniques, while 41% had adopted one or more agile development methods [2]. Another study was performed by a company called Version One in late 2006. This study was sponsored by the agile alliance and shows that as many as 84% of the respondents worked in a company that has adopted agile development practices anywhere within the organization. The latest survey had 722 respondents from 47 countries [24].

[2]Search done 25.05 2008, using Google Scholar

This heightened popularity is also reflected in the Norwegian development community, which hosted its first agile conference labeled *'Smidig'* in November 2007. The conference filled its capacity of 250 people with participants from all over Norway and consisted of approximately 90 presentations regarding experiences with agile in general and solutions to common challenges agile development faces.

Many of the agile opponents have argued that there is a lack of empirical data supporting the claims found in success stories. This is one of the reason why more and more research is focusing on this particular topic. A recent article published by Tore Dybå and Torgeir Dingsøyr featured a systematic review of the research done within the field of agile software development. This review identified 36 empirical studies on agile software development and found several challenges and benefits documented among them[10]. Even though the studies did not provide a unified view of the current practice, they provided a broad picture of experiences.

Another review done by Sridhar Nerur et al. summarizes agile development as an social interaction process where informal communication between the collective stakeholders provide the basis for action. The repeated cycle of "thought-action-reflection" ensures an adaptive and self-evolving environment. The diverse involvement of all team-members inhibits specialization and thereby increases diversity. This makes the team more equipped to respond to emergent situations.

Pekka Abrahamsson et al. concludes their review and analysis of several agile development methods with the following characteristics of the agile development process [18]:

- Incremental development (small software releases, with rapid cycles)
- Cooperative (customer and developers working constantly together with close communication)
- Straightforward (the method itself is easy to learn and to modify, well documented)
- Adaptive (able to makes last moment change)

Sridhar Nerur et al. argue that in order to respond to todays dynamic business environment an easy to adapt organizational structure is needed. This in turn require information systems that can evolve along with shifting requirements and the adaptive structure. They state that "traditional, plan-driven software development methodologies lack the flexibility to dynamically adjust the development process.” [20]. Their article review the challenges that an organization has to overcome in order to be able to take advantage of agile methods. While doing so they present a list over the most significant differences between agile and traditional/plan-based development methods. This list is reproduced in Table 2.1.

2.5 Agile motivation

The motivation for using agile methods is first and foremost to make the development process as efficient as possible. There are several potential benefits when using agile methods, but most of them have been questioned by professionals or the research community. This section summarizes the motivations found in the research literature. The sources for this summary is statements and experiences found in the research review by Dybø and Dingsøyr and the industrial study performed by Version One [24][10].

2.5.1 Customer collaboration, adapting to changes

A benefit found in the research review is the increased focus on customer collaboration. It reports that the customers were satisfied with the opportunity to give feedback and the quick response to change. This is in agreement with the industrial study, where 92% of the participants answered that switching to agile had improved the ability to manage changing priorities. This is probably the most agreed upon benefit that incremental/iterative development methods yield over plan-based methods, and is therefore one of the main motivations behind an agile shift.

2.5.2 Quality

Both the research review and the survey mentioned earlier also reports reduction of defects as a benefit. The review shows multiple experiences where the process for handling defects did have a positive influence on the results and the overall process. The survey showed that 75% of the participants experienced an improvement in software quality. These two experiences are not identical, but are probably caused by the same principle, namely the increased focus on continous integration and testing. This is indeed veryfied by the review

Aspect	*Traditional*	*Agile*
Fundamental assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management style	Command-and-control	Leadership-and-collaboration
Knowledge management	Explicit	Tacit
Role assignment	Individual-favors specialization	Self-organizing teams-encourages role interchangeability
Communication	Formal	Informal
Customer	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired organizational form/structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

Table 2.1: Differences between traditional and agile methods

where yet another experience reported less problem with system and integration testing. The increased focus on testing is not represented or required by all of the methods labled agile, it is best represented in XP and Test Driven Development (TDD), where testing during implementation is one of the main features.

2.5.3 Return On Investement (ROI)

Another reported benefit is that business value is demonstrated, and delivered more easily. This was reported by the research review and is most likely due to the cyclic lifecycle of iterative/incremental development. It makes sense that demonstrating new features during the project cycle enables the customer to easier understand what their requirements implies, and thereby facilitate continuous customer input. Short iterations between each delivery could also explain why 72% of the participants in the industrial survey answered that agile methods had improved *'accelarated time to market'*. Iterative development make it easier to evolve the software according to changing technological trends or priorities, and most projects deliver fully functional solutions between each iteration. It would therefore be easier for a customer to accept a partial delivery and use potential remaining iterations as updates or patches.

2.5.4 Improved control

The research review reports that project managers state that agile projects offer better estimation. The deviation from what each iteration actually used of resources and what was estimated, will then most likely depend on the size of the iteration, when each iteration is independent from the other. The review also found that another benefit was the way the management was forced to think ahead. Planning between each iteration forces the management to review plans on a regular interval, and might make it easier to spot difficulties, and estimate resource use. This is also closely related to the benefit reported by the industrial survey, namely reduced project risk. 72% of the participating companies found that agile methods reduced or significantly reduced project risk.

2.5.5 Morale and productivity

According to the research review the added focus on informal communication mechanisms combined with collective code ownership, standardization and tracking of progress was reported to help create awareness within the team. Developers also reported that they found that working agile made it easier to focus on current work, and thereby increased their efficiency. Another experience show that the developers working agile, were in all more satisfied with their working conditions. The survey confirms this by reporting that 74% of the participants experienced that the morale of the development team was improved. It also report that 75% found using agile development methods increased productivity, which suggests that the sum of the mentioned motivations can be benefitial.

2.6 Agile challenges

There are some areas where agile methods have been criticized or questioned by both practitioners and the academia. This section will review the challenges discovered in the research mentioned in Section 2.4.

While agile methods have some benefits that are indisputable, there also exist some challenges that cannot be ignored. One of these is to decide when they are suitable for use. Most of these methods rely on close collaboration, and are not easily adapted to distributed development environments, or development of larger systems, where the project team reaches a certain size.

Cohen et al. concludes his article called "Introduction to agile methods" by pointing out the need for rigorous communication mechanisms in larger projects. He argues that not all agile methods may be applicable in such settings because of their emphasis on verbal communication. He also argues that factors such as domain, criticality and innovativeness can impact the choice of development method. Life-critical systems that require strict quality control could also make "light" methods inadequate [8]. The review made by Dybå and Dingsøyr also report that the introduction of agile methods might be heavily dependent on the complexity of the target organization.

Abrahamsson et al. mentions the abundance of methods, and how it is more likely to cause confusion than clarity because of the inconsistent use of vocabulary and terminology. They also point out that there is an urgent need for more empirical data on the subject rather than more success stories or new agile models [18].

Pete McBreen makes several observations in his book titled ″Questioning Extreme Programming″. Among them is a statement that many of the claims made by the agile community are yet to be confirmed by scientific research. This could to some extent be explained by the recent introduction of most agile methods, but it confirms that the lack of empirical data is a concern. McBreen also argue that the lack of focus on architecture and design might from an traditionalist's viewpoint be ″an inefficient, questionable practice, possibly bordering on malpractice″. This, he argues, could thereby cause a project using agile methods to end up with a suboptimal design [17].

Chapter 3

State of the art

In order to broaden the understanding of how contract models affect an agile development project, a literature search has been conducted. The search is based on the technical report "Guidelines for performing Systematic Literature Reviews in Software Engineering" written by Barbara Kitchenham [14]. Due to the limited scope and the nature of this thesis, some of the stages in this guideline has been omitted. The documentation regarding the search is found in Appendix A. The results of the search include articles, white papers and experience reports that all are used as a basis for this thesis.

3.1 The buyer-supplier relationship

Companies in various industries become more and more dependent on information technology. They need tailor-made IT systems to suit their needs and gain an advantage in business. Many use professional IT companies to develop these systems for them, and it is a growing trend. A survey done by the analysis company TPI, showed that in the first six months of 2007, outsourcing contracts for 11 billions NOK were signed. This was an increase by 78% from the first six months in 2006. However, outsourcing the development of IT systems creates new challenges in form of a complicated buyer-supplier relationship.

A buyer-supplier relationship evolves over time. Tursas states that managing business relationships is not a linear process of moving the relationship in one direction towards an ideal state [21]. Ford et al. identifies four stages in a business relationship [9]:

- Pre-relationship

- Exploratory
- Development
- Stable

Learning, investment, adaptions, distance, trust and *commitment* are the concepts behind these four stages, and these concepts explain the development of a business relationship. Ford's model is useful for explaining the development of a relationship in the software industry. The four stages of a business relationship are represented in figure 3.1.

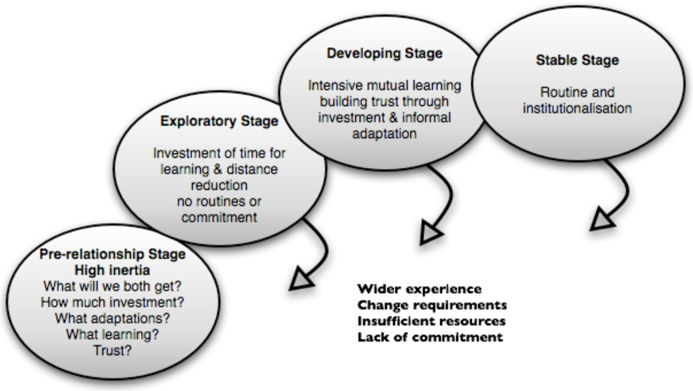

Figure 3.1: The four stages of a buyer-supplier relationship

The first stage, *Pre-relationship stage*, is all about contracting and forming the future relationship. According to Warsta this stage clarify what both parties get out of the relationship, how much they should invest in the relationship and how much they must adapt [25].

In the exploratory stage the parties negotiate a contract and exchange information. According to Tursas, the mutual learning is probably at its highest at this stage. Because of the insufficient knowledge about the other party, there is lack of trust on both sides.

The development stage may be reached after the exploratory stage. As mentioned earlier, the evolution of a business relationship cannot be viewed as a linear process towards a goal. In the development stage the different aspects of the relationship becomes more established. Contracts are signed and promises about adaption must now be realized. Based on the willingness to adapt, the commitment to each other grows, and trust is established. However, Wartsa states that too all-embracing contracts may indicate mistrust between the parties. This can take the relationship back to earlier stages in Ford's model.

When the stable stage is reached there is a certain stability in the learning, adaption and commitment to the relationship [25]. The cooperation and communication between the parties run smoothly because of the trust and the commitment that has been built up during the relationship.

According to Ford it is impossible to have a true agile relationship in the pre-relationship or the early exploratory stages of the relationship. Trust between the parties is a key issue, and in addition the parties must commit to each other.

3.2 The purpose of contracts

Contracts are used in order to reduce perceived risk between the buyer and the supplier. The buyer tries to ensure that they are getting most value for money, while the supplier must handle payment schedules, response times and pricing. These are concerns that must be agreed upon between the parties before the software development starts.

Poppendieck identifies two schools of thought for the purpose of contracts[16].

- *1. The purpose of contracts is to protect each party from opportunistic behavior on the part of the other party.*
- *2. The purpose of contracts is to set up appropriate incentives for companies to work together in a synergistic matter.*

Both of these two are related to trust between the supplier and the customer. When there is not sufficient trust between the supplier and the customer, one would likely try to prevent one another from opportunistic behavior. The contract model can affect the quality of the end result. On the other hand, if the customer have trust in the supplier, another (perhaps better suited) contract model can be chosen.

Tursas means that in an ideal situation, there would be no need for a written contract. An ideal situation is where the two parties form a close partnership and have no problems with information exchange [21]. Basically this means there are no uncertainties between the parties, and is again related to trust and the parties commitment to each other. Tursas also states you get more effect of reducing uncertainties the more all-embracing the contract is.

3.3 Agile development and contract models

This section will describe agile development in combination with contract models. Agile methods are embraced by many suppliers, but from a contractual point of view agile methods introduce some challenges.

3.3.1 Why is agile different?

In traditional software development methods, you use much time planning and estimating the system before you start developing it. You also have a requirement specification which define every feature and functionality of the system down to the very least detail. In addition, such methods often place much emphasis on documenting the whole process, from the requirement specification to design, implementation and testing. Traditional methods also often use a fixed price contract, where the requirement specification is used as a basis for the price.

Agile methods do not emphasize on using time developing detailed requirement specifications. The manifesto for agile software development clearly states that agile methods value *working software over comprehensive documentation.* Which means there is not nearly as much planning and estimation in agile projects as there is in traditional projects, and this is a challenge when seen from a contractual point of view.

3.3.2 Contract models

There are several different types of contract models which are commonly used. They are all characterized by their payment mechanisms which plays a big part in determining the way risk is distributed between the parties involved. The most commonly used are presented in this sub-section.

Fixed-price contracts [15]

When the customer and supplier agree on a fixed price that is to be paid to the supplier for the software it provides, it is called a fixed-price contract. This contract model places much risk on the supplier, since they have to complete the agreed upon software not exceeding the given price. This is the reason why it is the most commonly used contract in situations where little or no trust exist between the parties, or what Warsta calls the pre-relationship stage or the exploratory stage. If the supplier is not able to complete the agreed-upon software, they have to solely cover the remaining expenses to complete the software. This does not apply if the customer change the original scope of the software.

In this type of contract disagreements can easily arise between the customer and the supplier regarding if added functionality is within the scope or not. The supplier will profit from a change in scope, since this entails payment for added resources such as increase in time or cost. Combined with a traditional tender, bid situation, speculation on the part of the supplier may occur. This is most often described as underbidding, where the supplier make a low bid at a tender, knowing that changes most likely will make the contract worth while.

The price defined in this type of contract is most commonly found using estimation techniques. This means that the contract type suits traditional development methods well, since the requirement specification developed early in this development method can be used.

Time-and-Materials contracts [15]

A time-and-materials contract is a contract model where the customer pays the supplier by labor hours at a specified fixed hourly rate and material used. The hourly rates includes labor, overhead and profit. In this type of contract the customer takes most of the risk, and it therefore relies on strong bonds of trust between the parties. The supplier have little direct economical risk, but there will always be a risk of damaging their reputation by not delivering on time or delivering a low quality product. One might also argue that since the supplier is getting paid by the hour they have little incentive to complete the work, or at least a good incentive to use as much resources as possible.

Since this type of contract requires much trust between the two parties, it is not commonly used in situations where customer and supplier have little knowledge of one another, or in what Warsta calls the pre-relationship stage or the exploratory stage.

Target-cost contracts [15]

The target-cost contract model is a model which is used to share risk between supplier and customer. The two parties agrees on a price (the target-cost; which entails a fixed amount of hours they intend to use to complete the software) for a defined scope of work. If the costs of developing the software exceed the target-price, customer and supplier share the exceeding costs. On the other hand, if the supplier completes the software at a lower cost than the target-price, the customer will pay a certain amount (usually 50%) of the remaining hours.

This type of contract is similar to the fixed price contract in the way that any changes that are not in the original scope adds to the suppliers available resources. This makes it less critical to make an accurate estimation. The payment mechanism also creates an incentive for the supplier to complete the project as quickly as possible, and under the target price. Achieving this the customer would receive payment for work not done.

The sharing of risk helps this type of contract to be more accessible than the previous mentioned types. It can be used in all of Warsta's stages, with little or no adjustments from one case to the other. This combined with its reduced need for accurate estimation figures makes it well suited in agile development methods.

3.4 Norwegian contractual standards

There are two commonly used Norwegian contractual standards available to both the suppliers and customers. These are described in detail within this subsection.

3.4.1 PS2000

This is a contractual standard created by a research program called ”Project management 2000”. The research program consisted of a group of Norwegian ICT companies who worked together with NTNU and SINTEF. They started working in 1996 and released the first version of the standard in September 1999.

After the research program was terminated at the end of 1999, the Norwegian Computer Association (DND) took over the responsibility for further development and management of the contract. A second version of the contractual standard was published in 2001, and a third version in April 2007. PROMIS, a consultant company located in Oslo, are technically responsible and has written the contract details and guidances on how to use the contract [1].

The intention was to offer an alternative contractual model to situations when it is particularly difficult or unserviceable to draw up a detailed specification prior to tendering. It is the first Norwegian contractual standard which regulates implementation from iterative processes. The standard is revised periodical by a group of experts both from the customer and the suppliers side. The latest version was completed in January 2007, and the group are currently working on an addition that can be used as a guidance when using the contract with agile development methods.

Development method

The standard prepares for a stepwise implementation model, where the construction and composition of the software principally happens in iterative processes [1]. There are four main phases in the implementation model. The first phase is the *requirement assessment phase*, and it is written by the customer before the actual signing of the contract. In this phase, the customer has to analyse and specify their needs, purposes, and requirements which the project should cover. After this is done, the phase for preparing the description of the solution is started. The construction phase is the third phase, this phase is done using iterations. The number of iterations are defined in the contract.

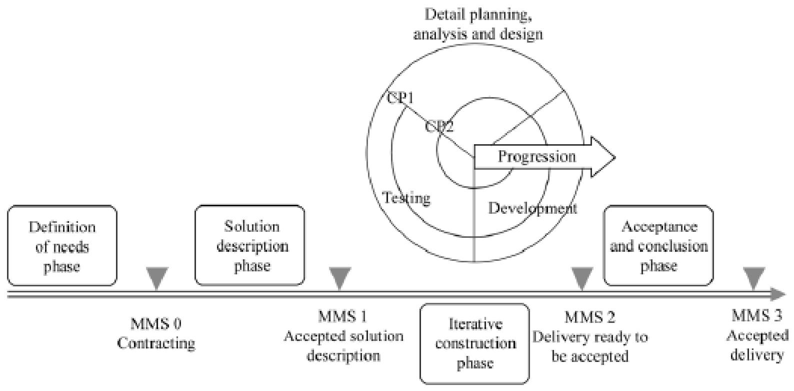
Detail planning, analysis and design
CP1
CP2
Progression
Testing
Development
Definition of needs phase
Solution description phase
Acceptance and conclusion phase
MMS 0 Contracting
MMS 1 Accepted solution description
Iterative construction phase
MMS 2 Delivery ready to be accepted
MMS 3 Accepted delivery

Figure 3.2: PS2000s model for implementation of software development project

After each iteration there is a checkpoint, where the customer will evaluate the implementation of the iteration and prepare a plan for the next step. Finally, when all the steps are implemented, there is an acceptance- and closing phase, where the customer conduct acceptance tests with support from the supplier. The delivery will be finalized with a project evaluation where both parties are obliged to participate.

Change management

Change management is described in the second part of PS2000. It states that only minor changes should be managed within an iteration. Larger changes should be handled in the checkpoints after an iteration. These changes must be delivered as change requests, and accumulate to the end of the current iteration.

When the iteration reaches the checkpoint decisions regarding whether or not they should be included are taken by the customer. The supplier is also obligated to issue a change request if the customer has inflicted work or cost which is not included in the specific delivery, or if items originally included in the delivery is left out due to time or resource constraints (a change request with negative scope).

The process of handling a change request requires an estimation done by the suppliers which is then delivered to the customer. The customer will then with the basis of the estimated consequences to the delivery, the price or the schedule decide whether or not this change will be accepted, and included in one of the following iterations.

Potential conflicts concerning a change order is solved by using the appointed steering committee, an independent third party or as a last resort the judicial system.

Customer and supplier duties

The customer is obliged to create an overview over their needs and requirements, with details concerning what effects these item should have, and how they can be tested after implementation. This is done prior to the signing of the contract, and is often used as the basis for the tender. After each iteration, at the checkpoints, the customer joins in the testing of the implemented features. At the end of the project, the customer is also responsible for running all acceptance tests with support from the supplier.

Before starting each iteration the supplier is responsible for a detailed plan regarding the next iteration. When an iteration is complete, the supplier is also required to present the work done in the iteration, the supplier is also required to deliver the agreed upon documentation, and educate the customers in the use of the product.

When the supplier have been selected the customer is required to participate along with the supplier in the development of a description of the wanted solution. This is only done prior to the first iteration and will be used as a guide through the following iterations. Both the supplier and the customer are required to fill out a risk-matrix, after a description of the solution is decided upon. This matrix consists of a list of foreseeable potential risks, where they all are evaluated and taken into account when pricing the project.

3.4.2 SSA-S

SSA is a Norwegian acronym and stands for ″The Norwegian Governmental Standard Agreement″ and contain a number of different contracts. The Agency for Public Management and eGovernment (DIFI) are responsible for the development of the contracts. They decided, after input from their user base, to adapt their software development contract SSA-U, where the U stands for ″utvikling″, the Norwegian word for development. The revision of this contract would make it more compliant to agile principles.

After starting the work DIFI realized that they needed to create an entirely new contractual standard, and called it SSA-S, where the S stands for ″smidig″, the Norwegian word for agile. SSA-S is currently under development and focus on making it easier for both the supplier and the customer to choose agile development methods in governmental projects.

SSA-S has borrowed some of the mechanisms from SCRUM and uses a variation of both the product and sprint backlog. It is based on short iterations of approximately 30 days and partial deliveries after each iteration. The customer must approve both the product and sprint backlog and has the opportunity to reject individual deliveries completely or partially [23].

While under development it is agreed that the following mechanisms are needed in order to ensure optimal co-operation:

- Guide and control the project in a manner that ensures the products quality
- Define and divide risk
- Prevent conflicts
- Resolve conflicts in the event that they occur

The development of this contract is done using draft documents which are sent to all users of the previously developed contracts, and people within communities engaged in agile software development. The last draft was sent October 10,

2007. A revised draft is expected any time soon.

3.4.3 SSA-U

The Norwegian Governmental Standard Agreement for Software Development (standard) (SSA-U) was created in 1999 as a part of a project called FASIT, where contractual details were identified as major element of risk in large governmental projects. The project was started in the wake of TRESS 90, a large governmental project that failed in 1995. It ended up costing approximately 2 billion NOK and 5 years of work. In 1993 The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology (NTH) (SINTEF) did an extensive documentation review and discovered that among other flaws, contractual details was interpreted differently by the projects stakeholders. [1].

Development method

The SSA-U was developed to ease the use of traditional development methods. This is ensured by dividing the projects delivery into three main phases. The specification phase comes first, it instructs the supplier to deliver an overall specification of the software, a detailed specification of the functionality and lastly a specification of the technical details.

The second phase defined in the contract is called the development phase and consist of the development of the software and the compilation of documentation. This phase also contains other eventualities like integration with existing systems, installation of the software and the education of potential users.

The third and last phase consist of acceptance testing, commissioning and delivery. The tests needed for the acceptance of the delivery would typically be produced along with the specification in the first phase and any disagreements such as errors or failed tests would be resolved using negotiation or an neutral third party.

Change management

The customer can, at any time, during the specification phase cancel the project and pay a predefined cancellation fee. After the first phase is completed the

[1]A presentation of TRESS 90: `http://www.idi.ntnu.no/emner/tdt4140/dokumenter/08-04-09-a.ppt`

supplier is also entitled to a percentage of the the total price as compensation. If the customer wishes to change the contract to either expand or reduce the scope, quality or execution of the delivery a request is sent to the supplier. The supplier is then obliged to perform an estimation of how this change affects other parts of the delivery and price. If the estimation is approved by the customer a change order is issued and the change implemented.

Any dispute in either the estimation or the change itself will be negotiated by either a liaison committee, a neutral third party or as a last resort the judicial system.

Customer and supplier duties

The supplier is obligated to deliver a solution that meets the requirements, the functionality and the overall objective of the product. The supplier is also required to at all times be able to supply up-to-date documentation of all aspects of the development.

The customer is obliged to give a precise description of the rationale behind their acquisition and their requirements. The customer is also responsible to make the conditions favorable for the supplier in terms of on-time deliveries, milestones or when other contractual obligation requires their cooperation.

Both parties are obligated to participate in supplier-customer meetings as long as a 3 day due notice of the meeting has been given.

3.5 Challenges with today's contract models

There has been some debate on suitable contracts for agile development, although there exists little research regarding this topic.

3.5.1 Agile development with fixed-price contracts

Some attention has been given to whether fixed-price contracts can be combined with agile methods, and there are different opinions. Martin et al. investigated two projects where XP was used, where one project was on a fixed-price contract while the other was on a time-and-material contract [3]. They found that the project using fixed-price contract clashed with the XP process.

Schah states several reasons for why fixed-price and fixed-date contracts does not work well with agile development [19]:

- Hard to estimate in a purely agile project
- Estimating time and cost is time consuming, thus taking away much of the benefits an agile approach would bring
- Fixed-price, date or scope pitch the customer and the supplier against one another.

The website *Agilekiwi - Practical Agile Software Development*[2], which is concerned with *connecting agile development with customers*, agrees to some extent with Schah, but is more optimistic. It says fixed-price contracts are not necessarily the best option for an agile project, and you may not get all the benefits of agility. But as long as the requirements are stable, it can be done. However, if the requirements are not stable, it is advisable to use another contract model.

Fowler is very negative to using fixed-price and fixed-scope contracts in agile projects [12]. He says you can very well use a fixed-price contract in an agile project, but you cannot have a fixed scope. Many companies embrace contracts with fixed-price and fixed-scope since they think it will lower their risk. If they do not get satisfactory software, then it will not cost them extra. This is something Fowler calls the *FixedScopeMirage*. Focusing only on the cost of the software is short-sighted. The software has a business value, a value that is greater than the cost. So if you do not get satisfactory software, you loose the business value the software was intended to create. Fowler also points out that a fixed scope contract is only fixed if the supplier really understands the requirements, and argues that this rarely happens.

Because of these reasons, Fowler discourage the use of fixed-price and fixed-scope contracts in agile projects. Poppendieck et al. has included a statement from a former software development manager which supports Fowler's arguments [16]. The manager explains that he ran a software company which prided itself in not exceeding the price and schedule agreed upon at the beginning of the project. Over a three-year period, 77 of 78 projects was completed on time, on budget and within scope. The problem was that none of the customers were completely satisfied.

Wingård also agrees with Fowler [26]. He states that it is not possible to have a complete picture of how a complex IT-solution best should function in advance. You learn more about the needs and what will be a good solution through the

[2]http://www.agilekiwi.com

development process. This applies both to the supplier and the customer. Instead of specifying every functionality of the system and having a fixed budget, Wingård argues that it is better to just have a basic idea of how the software should work and how much money you are willin to spend on it. This way, it is easier to manage requirement changes during the development process.

3.5.2 Agile development with other contract models

There are other contract models that are more suitable for agile projects than fixed-price, and the most common is target-cost and time-and-material contract. These were described earlier in this chapter.

The time-and-materials contract is perhaps the best suited for agile projects. It can however be difficult to get the customer to approve this kind of contract, since they are then left with most of the risk. Eckfeldt et al. wrote an experience report on the use of target-cost contracts in an agile environment, and they found that it is hard to sell time and materials contracts to the customer. The customers opinions were that *"time-and-materials is favorable in small budget projects and working with a consultant in which you have a high degree of confidence."* For larger projects or when working with unknown consultants the customers preferred fixed-price contracts. It is easier to sell a target-cost contract, since it shares the risk between the supplier and customer. The customer, in the case presented, gained trust in the supplier when using this contract model, because he felt that *"they were in this together"*. Eckfeldt et al. says that the target-cost contract model was well suited for agile projects, although the contract model handles changes in scope no better than a regular fixed-price contract.

Trust is consequently an important factor when it comes to software development contracts. Poppendieck and Poppendieck supports this statement [15]. They describe how Toyota has had success by gaining its suppliers confidence. Toyota understands that a strong supplier network is far more beneficial to its interests than the short-term gains that comes from taking advantage of a supplier.

Trust is especially important when it comes to time-and-materials contracts, because it gives the supplier a possibility to take advantage of the customer. It is also important for target-cost contracts, although this contract model divides the risk more evenly between the supplier and the customer.

Chapter 4

Research

This chapter contains an overview of the research done in this thesis. It is divided into six parts covering the the description of the goal of the research and the research questions. It continues to review the considerations made when choosing research approach, and how the research is designed. It also describes how the research process proceeded, and lastly this thesis' limitations and biases.

4.1 Research goal and question

The problem definition is as mentioned in Section 1.3:

As more and more ICT-companies migrate to agile methods, the existing contracts and contractual frameworks are mostly tailored for plan-based development methods. The goal of this research is to discover what experiences the Norwegian industry has regarding this issue, and how this is dealt with.

From this description two research questions was extracted, that is later addressed in this thesis. These are:

1. What are the experiences with using agile methods in practice and how does the contractual model used in a project affect its ability to use such methods?

2. What work is being done to help adapt contract models and standards to agile methods in the Norwegian ICT-industry?

4.2 Research approach

Research within the field of software engineering have three commonly used approaches; case-studies, experiments and surveys. A *case-study* is most often used to monitor or observe already existing or the introduction of new processes, tools, methods etc.. It usually includes a comparison to measurements called the baseline, this comparison is used in order to measure the effects of the process, tool, etc. An *experiment* is a controlled investigation conducted formally and with very specific constraints. It makes it possible to control all the independent variables that might interfere with the results, making them easier to interpret. A *survey* is an investigation, usually performed in retrospect by analysing collected data, most often collected by the use of interviews or questioners.

The research approach used in this thesis started out as a case study. More exactly, the research was intended to be a case study of the Norwegian ICT-company Ciber. The intention was to gather views on agile methods in combination with various contract models from people on different levels in the organization. After the first contact with Ciber it became clear that a case study with only one company would not yield sufficient results, and that the research needed to be expanded. The method used would therefore best be described as a survey.

According to Claes Wohlin et. al., a survey can either be descriptive, explanatory or exploratory [7]. A descriptive survey is where the research questions are already known and the survey tries to describe a situation or characteristics of a population. Explanatory surveys are used when one tries to find the relationships between observations. Exploratory surveys are used when the research questions are not clearly defined, and often performed when the topic is not clearly understood. This thesis fall into the last category, and several revisions of the first research questions has been performed.

There are two main types of research paradigms that have different approaches to empirical studies. *"Quantitative research is concerned with studying objects in their natural setting"*, and is in the case of surveys represented in the form of an interview [7]. *"Quantitative research is mainly concerned with quantifying a relationship or to compare two or more groups"*, and are in cases of surveys done using questionnaires [7]. The difference between these two are often said simply to be the collected data, with qualitative research the data returned is text, and with quantitative research the data returned is numbers.

Since it was discovered that fewer people than anticipated had knowledge regarding this subject, it would have been difficult to get responses from a population large enough for an reliable statistical analysis. It was also considered to be the easiest way to make use of the existing connection with Ciber. The research can therefore be classified as a qualitative.

4.3 Research design

This section will further explain the reasoning behind the structure and design of the actual research performed. It will also take a closer look on the selection of the sample used and present some basic information about the companies included in the sample and the details of the companies that ended up participating.

4.3.1 Interview

The interview was designed based on the research questions and the information gained during preliminary communication with Ciber. It was decided to use a combination of a general interview guide approach and a standardized, open-ended interview in order to ensure an easier analysis and a speedier interview process [22]. In order to do this an interview guide was created which gave a quick overview over the topics that should be covered and included some opening questions that could help the respondents get started. Most answers to these questions would make it easier to continue with follow-up questions that might not apply to all the respondents and therefore not be as easily put in the guide. The answers received in the first interviews also helped us get an overview over topics or experiences that were unfamiliar to us, and therefore added to later guides.

The interview guides can be found in Appendix B.

4.3.2 Interview subjects

At the first visit to Ciber a number of system developers and project managers of two different projects were interviewed, in addition to the head of the system development. After collecting and organizing the data, it became apparent that the developers had little knowledge and few opinions concerning contract models. The developers were mostly concerned with their day to day workings and the agile techniques used in the project, and left the administrative concerns to the project manager. The project managers did have some interesting experiences, but most of the contributions came from the head of system development.

After realizing where the target subjects most likely were situated in an organization, it was decided to expand the research to also include other companies and people with experience with customers, sales or contract negotiation. Information regarding current work on Norwegian contractual standards was also discovered during the first interviews and the literature search, and it was decided to include this aspect in the research as well. The interview subjects is therefore divided into two separate groups. One group represent practitioners with industrial experience in using agile development methods (marked as industry in Table 4.1), and the other group are experienced with creating and adapting contracts used in the industry (marked as contract in Table 4.1).

When considering which subjects we wanted to interview, it seemed like choosing a convenience sample was the only option. This decision was reached on the basis of the timescale this thesis is written, and the limited number of people with the experience needed. A convenience sample is an non-probability method used for picking respondents. It is most often used as an inexpensive estimation of the result where, as the name implies, the most convenient sample is picked [7]. The interview subjects were therefore selected based on input from either our co-supervisor Geir Kjetil Hanssen, the first interviews performed at Ciber or names discovered during the literature search.

Name	*Role*	*Company*	*Date*
Marianne Selle	Industry	Ciber	14.03.08
Sverre Tinnen	Industry	KnowIT-ObjectNet	15.04.08
Lars Ivar Næss	Industry	KnowIT-ObjectNet	15.04.08
Mari Vestre	Contract	DIFI	16.04.08
Kjetil Strand	Contract	PROMIS	17.04.08
Jørgen Petersen	Contract	PROMIS	17.04.08
Reidar Strand	Industry	BEKK	21.04.08

Table 4.1: Interview subject, their company and the interview date

Table 4.1 contains the list of interviews performed after the revision of our research goal. In addition our first round of interviews were performed on 7 developers and 2 project managers from two different projects. These are not included in the list, since the projects where they were interviewed is are requested to keep anonymous. The other interviewees had also an option to be anonymous, but chose not to.

4.4 Research process

All the respondents were first contacted with an e-mail describing the topic of our thesis and how their contributions would potentially be of use. The e-mail did also include the interview guide and an explanation of how the interviews would be performed. The e-mail was followed shortly by a phone call in which further details were described and a date and time suggested. All of the participants approached agreed to do an interview.

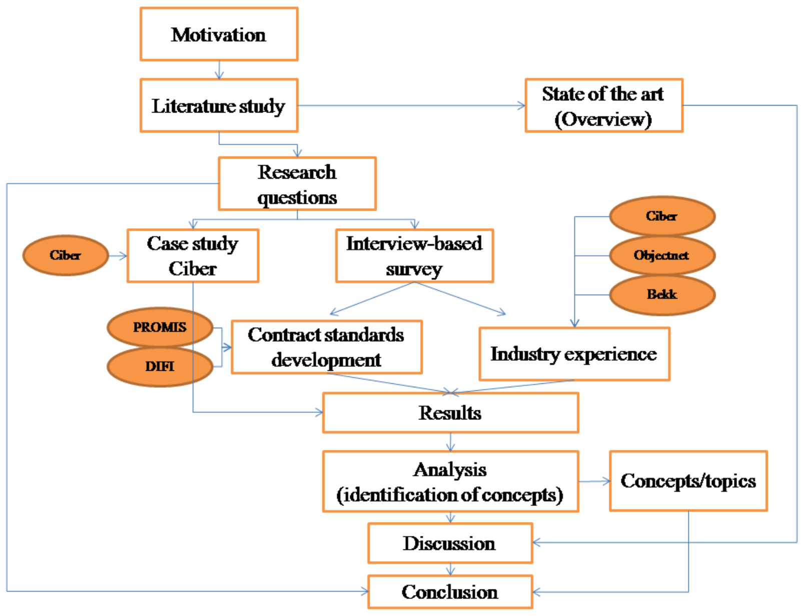

Figure 4.1: The work process of this Master's Thesis

Prior to all interviews all participants were asked whether or not they minded that the interview was recorded, which they all consented to. All but one interview was performed with both authors present, were one had the role as main interviewer, and the other as minutes secretary. The first interviews performed at Ciber were performed at the 13. and 14. of March, the remaining interviews were performed at the dates listed in Table 4.1. All of the interview subjects were located in Oslo, and since it required traveling and overnight stays, most of the interviews were done in within a period of 2 and 3 days. The time used on each interview varied from approximately 20 minutes to 2 hours, the shortest interviews were the ones done with developers.

After the interviews were completed the summary used in Chapter 5 were sent to the interview subjects for approval. None of the summaries were rejected and the results was used further in Chapter 6. An overview of the overall work process, and how the different parts of the thesis is used is presented in Figure 4.1.

4.5 Limitations

One limitation of this research is the lack of customer input. There are no interviews of product owners who normally participate in the project team, and who would undoubtfully contributed to this research. Input from customer representatives who are involved in buying and the negotiation of contracts would also contributed with a viewpoint of this topic which is not currently represented. Mari Vestre do however contribute with some insight on this subject, having consulted customers in both contractual negotiations and the development process. This is a deliberate limitation, and due to the limited time and resources available.

The number of interviews also prevent this thesis to result in any statistical valid arguments. One might also argue that the participants chosen for the interviews do not represent the average Norwegian supplier, and that the results could therefore differ if another population had been chosen. The consideration done by the authors however is that there is a limited number of industry subjects with knowledge about both agile development methods and their use under different contracts, and we therefore assume that their experiences are close to accurate and a fair representation of the state-of-the-practice.

4.6 Bias

All of the interview participants would be inclined to portray their own organization the best way possible. This could influence their responses, and cause them to downplay problems that in reality are quite severe. One can also argue that neither the developers behind PS2000, Kjetil Strand and Jørgen Petersen, or the developer of the new SSA-S, Mari Vestre, can be regarded as neutral.

Due to the limited literature available on the subject the research is also heavily dependent on few sources that are difficult to verify.

Chapter 5

Results

This chapter will present the results of a series of interviews made in the period between march and April 2008. It is not a word for word transcription of the interviews, but a summary that will help improve readability.

The results are divided into two main parts where the first contains the results of the intervies done with industry practitioners and the second interviews done with what can be described as contract experts.

5.1 Development managers

5.1.1 Sverre Tinnen

Sverre Tinnen starts by explaining that Objectnet has worked evolutionary for a long time, and has in recent years been pulled towards agile development methods. Today agile development methods are widely in use. He explains that they have spent much time adapting their development methods to suit especially rigid fixed-price projects, or projects that use SSA-U. In order to make agile methods work in these settings, a few creative adaptations are needed, he says. As an example he mentiones the specification phase of SSA-U, and how it releases the first payment when it is completed. This phase is all about specifying details in the planned solution and handing it over to the customer, since a complete specification early in the development is in conflict with agile principles this has to be circumvented. Sverre says that they do this by working with the customer to define a start-up phase with deliveries that they both agree should be finished early on. When this phase is completed the payment is released. Sverre says that in their experience traditional contracts do not hinder the use of agile methods. He explains how it can make things more difficult in an introductory sale, but most aspects of the actual method can be attached to the contracts and does not necessarily pose a problem. He states that the waterfall process is no more, and it is up to the suppliers to find alternatives that can be accepted as documentation.

He has experienced projects where user stories together with proof of concept was delivered and accepted as architectural design. When asked how the customer reacts to what is perceived as inferior documentation quality compared to traditional documentation, Sverre explains that most of their customers appreciate the suppliers expertise in the area and therefore accept the guidelines given on how the development process can be efficiently managed. He also states that it has become more and more common to outsource the management of a project, and that in these cases a more formalized process is needed. Sverre argues that handling conflicts due to requirement changes also needs a certain degree of formalism in order to avoid conflicts. Documenting requirement changes and making the customer aware of both the estimated price and consequences is highly important, he says. He goes on to explain that if the project is run under a fixed price contract, it is important that the customer is forced to exclude something for each added functionality. One way to achieve this is to have a meta system that is formal enough to be accepted in cases where potential conflicts arise. This system can host summaries of meetings and decisions that affect the contractual obligations. These seldom surpass the original specifications or requirements, but might resolve minor conflicts without sacrificing the agile aspects.

The biggest problem in modern software development is probably not the contract in any case, Sverre states. A real and responsible involvement from the customer is difficult to achieve even though agile development, with mechanisms that encourage communication between the participants, has given us a nudge in the right direction. The most important thing agile methods has contributed with is collective responsibility.

Both the customer and the supplier knows that what the contract specifies is neither what you want to end up with or what you are going to end up with, Sverre explains.

When asked how customer involvement is handled, Sverre answers; *"we do experience that neither contract or customers directly oppose the use of agile development methods. The customers are often indifferent to the development methods. They want a modern and sensible development process, and is easily persuaded to use agile methods,* *but they rarely know what this entails. This results in a product owner that is more often than not a hostage."* Sverre goes on to say that in addition to rarely having truly involved customer representatives, another problem is lack of authority or ability to make a decision. A truly actively involved customer representative is rare. Customer involvement is time consuming and most contracts do not handle this aspect. He continues to say that the new SSA-S focuses on the distribution of responsibility, but that there is still a long way to go before contracts are were they need to be, that is when customer involvement is regulated by contract.

Sverre argues that the actual sale of a project starts after you have signed the contract. So it is then you start to sell the *"agile"* package. The challenge is to find a solution that solve the problem in such a way that it is economically beneficial for the supplier and at the same time as good as possible for the customer. It is important to perform this assessment often in order to ensure a satisfied customer and a profitable business. In this regard Sverre also mentions the importance of involving as many of the users as possible in order to have them argue the feasibility of the solution when it is needed. 'Friends' within the customer organization also help resolve any possible conflicts that may arise.

You need to, given the contractual conditions, ensure the best possible solution, Sverre says.

Sverre says that they have limited experience with PS2000. He explains that in their experience the real challenge of PS2000 lies in the first phase where all the details are worked out. He adds that he finds it strange to finalize the details this early. He also explains that the model is quite comprehensive, mapping both risks and the distribution of responsibility early on, but he does not find that it stimulate real involvement more than any other contract. He also adds that in most settings the ability to restrain from having to review the contract is more important than managing all aspects of a project contractually. One way to achieve this is to make sure that the customer knows that you have their best interest at heart, and thereby develop a relationship based on trust. This is easiest done by maintaining a good track record with the customer over time.

5.1.2 Reidar Sande

Reidar Sande is the chief operating officer at Bekk. This means he is responsible for all the projects run by Bekk.

He says he has used agile methods for some years now and he has good experience with it. He has used agile methods with many different contract models, including fixed-price contract. The fixed-price project was a success, but this was only because there was a good relationship of trust with the customer. There are two things in a fixed-price project which disagree with agile methods, he explains. Firstly, there is much upfront work, i.e. planning and estimating. Secondly, you have the change management process, which can be very bureaucratic, and therefore conflicts with agile principles.

He explains that when you have to combine fixed-price with agile development, it is because the customer has no trust in the supplier or they can be on a tight budget. The customer feel like they are eliminating some of the economic risks by having a fixed-price contract. But it is not ideal to run an agile project with a fixed-price contract. He thinks the customer gets less value for money. First of all, there are a lot more conflicts regarding changes in a fixed-price contract. Secondly, the customer strive to get most out his money, while the supplier will have a least effort attitude. So there is an ongoing conflict between the parties through out the lifecycle.

But he states there are also positive aspects by having an agile fixed-price project. You create simple solutions. He says that they have a saying in his company: *"Don't develop a Rolls Royce if the customer wants a Lada"*. The opposite can happen when you have a time-and-material contract, he says. You want to show off your expertise and create a fancy system, with lots of neat features. But the customer did not ask for this, and is certainly not going to pay for it. So it important to remember that requirements are to be met, and nothing beyond that. And this is much more ensured in a fixed-price contract than in a time-and-materials contract, he states.

It varies on how the customer agrees on doing an agile project. Sometimes the customer themselves want it, because they have heard of it and "'everybody else is doing it"'. But not all customers really get agile methods. He says as a supplier they are responsible for educating the customer. Agile methods do not work without the customer, that is a fact.

He likes agile methods more than the traditional waterfall development method. When using agile, the customer get something they need and want. This is not always the case if you use the traditional development method. And it is few problems with the system- and integration test, because the system has been thoroughly tested through out the implementation process. But agile projects can also fail, he states. The difference from traditional methods is that in an agile project you can see a possible failure in advance. It will not come as a shock, as it tend to do with the waterfall method.

He was involved in the development of the PS2000, but not any more. Currently a guidance to to the PS2000 is developed. The guidance is object how to use agile methods with PS2000. PS2000 can be interpreted as both a traditional and an agile contract. By publishing the agile guidance, people will get a better understanding of how they can use agile methods with the contract model. The guidance will also help the customer and the supplier to get a mutual understanding of the contract. Then you avoid that the customer have the waterfall-interpretation of the contract, while the supplier have a agile-interpretation, he explains.

He argues that the current SSA-S is a hundred percent waterfall oriented. If he could choose between the SSA-S and PS2000, he would choose PS2000. But he explains that something must be done in both SSA-S and PS2000 when it comes to sanctions about day fines. The most ideal contract model for agile methods as he sees it, is time-and-materials. This requires trust between the customer and supplier.

He says you can use PS2000 with agile methods. You make an initial description of the solution where you set up the number of planned iterations. Then you evaluate after each check point. He explains if they then discover that they need an extra iteration to finish the solution, they have to tell the customer. The customer can then choose to accept this, or remove features from the scope. He says that the length of the sprints depend on the time it takes for the customers to get clarification on different issues and reach decisions. He says he has not noticed any difference between governmental customer and customers from the private sector when it comes to how long it takes to reach a decision. But he adds that PS2000 is not an agile contract, although you can combine agile methods with it. You need a description of the solution, and here it depends on how you define this description. You can describe the system down to the very last detail, or you can make a coarse sketch. If you do the last one, you can work agile with PS2000, he explains.

He says you must have control on all the documents when working agile. Often, you deliver a system that is not pursuant to the original requirement specification. You will then get into trouble if you have not documented why some features are left out and why some new features are included. You must document all changes that are made from the original requirement specification, so the customer can see when and why all decisions was reached. These documents will outrank the requirement specification if it is specified in the contract. If you do not document changes and decisions made during the development process, the customer can refer to the requirement specification and complain about why they have not got the system specified there.

When asked about conflicts regarding changes, he says he has often experienced this, and he argues that such conflicts is something that comes with software development. The customer and supplier often disagree whether a new feature is within the original scope or not. The customer's functional responsible and the supplier's functional architect will then get together and try to resolve the conflict. If they do not manage to resolve the issue, the conflict will be escalated to the project managers. If an agreement is not met, the conflict will be addressed in a steering committee. This however happens rarely, he says.

5.1.3 Marianne Selle

Marianne is the head of the software development department at Ciber. That means she is responsible for all offers delivered by Ciber, and is also responsible for all the development projects run by the company.

She says she prefers agile development methods over the more traditional approaches, because that way it is easier to deliver a system that the customer really needs. And she says she likes the frequent deliveries, because then the customer has something hefty to deal with, and can more easily come with change requests. All new projects run by Ciber use SCRUM. But there are however a few show stoppers when it comes to agile methods, she explains. If there is no trust between the supplier and the customer, working agile becomes very difficult. Trust is everything, she states. Also, the customer must know what they are getting into. If the customer does not know what is expected of them, it is hopeless. She also points out that if a project wishes to succeed with agile methods, the project team must have an agile enthusiast. You cannot impose that a project should use agile methods, if no one in the project team really wants to. The project team must have a driving-force who promotes the use of agile methods.

She says governmental customers use SSA-U as if it was the law. SSA-U prepares for a fixed-price, a fixed scope and a fixed time schedule. This is a challenge if you want to apply agile methods. A great challenge with a fixed delivery date is that it conflicts with the change management process. It takes time to elucidate a change, and resources used on actual development are absorbed by the elucidation process. SSA-U has no controlling mechanisms for this aspect, and it becomes a problem when the customer does not realize that elucidation of changes will affect the agreed upon delivery date, she explains. She says Ciber's projects only use Scrum's reporting mechanisms when they have projects based on SSA-U.

If customers feel innovative or has hired external consultants, they use PS2000, she states. But PS2000 has generally little attention and impact in public organizations. PS2000 is much better than SSA-U when it comes to the planning phase. But the contract model still has some of the same problems as SSA-U. For instance, there are no good mechanisms for managing changes. The check points in PS2000 are also a problem, because it is too much work going through with a check point every four weeks. She feels the contract model is a little too formal. She says she has used PS2000 on a few occasions, and she rarely experience that the customer really understands the contract and what is expected of customer. PS2000, by her opinion, only promote a lenient form of agile development, and still the customer do not get it. It is hard for them to understand the degree of involvement that is required of them. But despite that she thinks PS2000 is the best contract model today.

Customer involvement is Alpha and Omega, she says. But it is a challenge to get the customer to be in on it. And the customer has to dare to do an agile project, which is not always the case.

The customers want a price on the system they are buying. But it is hard to make precise estimates. Some customers do not understand this, and she explains that this has something to do with the age of the customer's decision makers. They are usually over 40 years old, and information technology was a different world the time they went to school. She argues that the complexity of today's IT systems cannot even be compared to the system back in those days. Web services and Service Oriented Architecture (SOA) makes the today's systems very complex. She says that some customers think that SOA makes it easier to estimate the cost of the system, but this is not right. She says it is a challenge to explain to the customers that SOA makes the system and estimation more complex.

She says she prefers framework agreements. All the risk will then be with the customer, since it's they who are running the project. All the supplier has to be worried about is the performance of their developers. You also avoid the conflict between the supplier and customer, since the supplier now only hire out competence and is not in charge of the project. She notices that such framework agreements are more and more used, and the traditional method where the customer invites tenders, is decreasing. But she concludes with that the contract is only a guidance. It is something that is put away immediately after it is signed, and never looked at again, unless a conflict between the supplier and the customer arises. She says it all boils down to trust. The biggest show stopper for agile development is the lack of trust between customer and supplier, she concludes.

5.1.4 Other sources

In the first interview round, project managers and developers in projects Ciber were involved in were interviewed. This included project managers and developers both from Ciber and Bekk. As the focus of this report took a turn in another direction, described in Section 4.1, much of the material from these interviews were no longer relevant. This especially applies to the interviews with the developers. However, some of the statements from the interviewees are relevant, and this section sums them up. In order to ensure the anonymity of the projects, the names of the project managers are not included.

Project Manager, Bekk

This project manager has no negative experiences with agile development, but she says that it is a challenge to teach the customer how to work agile.

She talks about her current project, and says agile methods has worked very well in the project team. The project is behind schedule and is over budget, and she means that this is related to that the product owner is not in control of the budget. So the product owner just request features he thinks should be included in the system, without thinking of the economical consequences. She states that when working agile, it is very important that the product owner is also the budget owner.

When it comes to contracts, she means that the development method should be independent of the contract type. In this project they currently have a fixed-price contract, but they started out with a target-price contract. The reason why the contract was changed from target-price to fixed-price was that the customer felt that they had all the risk, and requested therefor a fixed-price for the remaining work. She says that this results in a much stricter change management process. In addition, she has told the members of the project team to be more aware whether features requested by the customer are inside or outside of the current scope.

Project Manager, Ciber

This project manager means that the lack of documentation by the developers makes it more difficult for him to get a detailed view of the work done other than the issues discussed at the stand-up meetings. He also states that the lack of planning and reduced perspective can lead to work being done twice.

His preferred contractual model would be target price where one would work as if using time and materials, and then divide the potential extra expenses and costs. He also argues that in most situations it is hard to defend using the fixed-price model, since it requires accurate estimation, which is very hard to do.

5.2 Contract experts

5.2.1 Jørgen Petersen, Kjetil Strand

Jørgen starts by presenting PROMIS and their field of work. He explains that their main area of interest is project management and the creation of development methods. PROMIS was technically responsible for a research project called 'Project Management year 2000'. The project ended up with a list of best practices on how to manage a software development project. They decided that creating a development method to rival the big ones would be difficult to achieve, and therefore decided instead to incorporate their best practices in a contractual model called PS2000. A lot of their work revolve around PS2000, but they let The Norwegian Computer Society (NCS) manage the standard since PROMIS do not have the manpower to do it themselves. Jørgen also says that their efforts lately have been to replace PS2000 and its original connection to Rational Unified Process (RUP) and DSDM with a new connection to agile methods. Kjetil adds that even though they have done a lot of work with RUP, they have never used the entire framework, which he characterizes as rather voluminous. Their experience is therefore mostly with variants of RUP, which is what they mapped PS2000 onto from the start. He continues to explain that there exist research that focuses on defining the differences between RUP and SCRUM, and that a lot of this research points towards customer involvement as the biggest difference. It is possible to manage a project using RUP and complete customer involvement, but it is not supported by the original description of the method, Kjetil says. Jørgen adds that this is one of the areas that PS2000 complement the development method itself.

Recently PROMIS has been involved in the work of improving PS2000 and creating guidelines on how to use it in an agile project setting. Kjetil says the guidance address two main user concerns, one is the concern customers have on whether or not it is possible for them to use agile methods, and the other is aimed at agile developers who are skeptical of using fixed contracts in an agile setting. Jørgen states that both are barriers that need to be overcome and adds that another aspect of these guidelines is to make the connection between

PS2000 and agile concepts official. He says that even though the 'agile' is very popular, there seems to be little in the way of instructions on how to manage it administratively. There is a difference between religious agile supporters and skeptics Kjetil adds. It is therefore important to inform the customer where agility is suited and offer an alternative where it is not.

When asked whether they had negative experiences with other contractual models, Jørgen states that he finds the current popular models too narrow minded, and angled towards the waterfall development method. He adds that there is lack of guidance that might have helped to simplify the process. Previous experience also shows that models where development is done iteratively, the participants focuses on communication and have a common incentive have a higher success rate. Kjetil adds that in addition to these experiences PS2000 focuses on risk-sharing, he also argues that the SSA-U is customer centric, leaving the supplier with most of the risk. He goes on to explain that the use of incentives in PS2000 makes it possible to model the pricing mechanism to be the midway between fixed price and time and materials. This is done by rewarding both the supplier and the customer if the price is below the estimate and by paying 50% of the efforts beyond the target price.

Jørgen adds that in addition to the incentive model, there are other mechanisms that help divide the risk. The option to cancel the delivery after the first phase where the solution is described, enables the customer to either completely abort the project or continue with another supplier. In this phase the participants are also required to fill out a risk-matrix, where all risks and measures as to how they would be dealt with is presented. This, Jørgen argues, ensure a close collaboration. Kjetil says that foreign customers that are faced with the English version of PS2000 react with skepticism to this notion of revealing their own risks. Their fear is that the supplier will use this knowledge against them, when in truth it is the quite the opposite, risks that are exposed will be easier to deal with using measures agreed upon, and possibly avoid financial losses. This is one of the reasons why PS2000 has received attention internationally.

People in the business are warming up to the model, Jørgen says. Mari Vestre talks positively about the contract, but this could be because there lacks an alternative. Kjetil states that her view is that PS2000 is an iterative contract, while SSA-S will take the next step and be an agile contract. Some of the details in SSA-S are based on the current version of PS2000, and neither Kjetil or Jørgen have discovered any substantial innovation in the available drafts. Both Kjetil and Jørgen agree that lawyers have had too much of an influence in the previous contracts made available from DIFI. Jørgen explains that they try to let engineers with experience develop the contract, placing the lawyers in the background. Kjetil argues that getting input from participants that supply the solution is important, and adds that he is worried that not enough of input of this sort have been included in SSA-S. Kjetil states that of what he has read so far SSA-S there seem also to be a few shortcomings, among these is how exactly the partial deliveries will be rejected or accepted. It seems that it is unavoidable to end up with a large amount of acceptance testing after each delivery, a drawback that will impede agility. He also adds that this particular problem is addressed in the guidance soon to be available for PS2000. This is indeed one of the major issues they have had to deal with when adapting the model to SCRUM, Jørgen says. Moving from longer iterations towards iterations as short as a month has been a major challenge, but they still feel it is the right way to go, not only to make it resemble SCRUM, but to ensure as good a process as possible.

Jørgen says: *"We are of the opinion that PS2000 with the guidance under development will be sufficient, and the best alternative to an agile contract"*. He adds that despite the fact that both SSA-S and PS2000 have the same customer base, there seems to be little migration from one to the other. They explain that there seem to be a slight difference between governmental or privately owned companies and the contractual preferences in the sense that private customers seem to value negotiation efficiency. There are companies that have extensive experience with SSA-U, who gain financially by being highly efficient in the start-up phase of a project. At the same time there seem to be a growing base of suppliers that value PS2000. When asked about PS2000s focus on customer involvement, Jørgen says that it is an aspect that are of crucial importance to ensure success, and in order to benefit from, some pitfalls need to be avoided. Foreign participants in particular often refuse to include this part of the contract based on the extra risk it puts on the customer. Kjetil adds that this part of the contract can be perceived as a benefit to the suppliers since any contractual breach would cause and extension or increase in price. The reward when the customer is willing to take this risk is a much close collaboration which most often leads to a superior end result. Kjetil concludes that the biggest challenge with using agile methods and PS2000 is the contractual foundation. To think you can specify everything from the start will lead to a suboptimal process. There is, on contrary to popular belief, less risk in under specifying.

5.2.2 Mari Vestre

Mari Vestre works for DIFI. DIFI aims to strengthen the government's work in renewing the Norwegian public sector and improve the organisation and efficiency of government administration. They have made a numerous of different contract models for various purposes available, and Mari Vestre is one of the responsible persons for these contracts.

She started to develop a new version of the SSA-S to try to modernize and be up to date. She has always been interested in agile methods. The idea is that it should be possible to combine agile development with the new version. It's supposed to be published sometime in the summer 2008. The goal is to get the contract as close to agile methods as possible.

She explains that the current version of the contractual model does not define a development method, but still many people claim it is meant for the traditional waterfall method. A great challenge with developing the contractual model is that you are bound to follow the governmental directions, and this limits the freedom to act.

She says that the problem in a contract is that you always have to regulate responsibility. She likes to compare distribution of responsibility with a relay race. First the customer has the baton, i.e. the responsibility, and then gives the responsibility over to the supplier, and then it goes back to the customer again. And it continues like this through the entire project lifecycle. In an agile setting, you have to share this responsibility, she says. But at the same time, you must have an unambiguous distribution of responsibility so you can establish whose fault it is if something goes wrong. In a fixed-price contract, all of the responsibility lies on the supplier. In a time-and-material contract, the responsibility is on the customer. The goal is to find something in between, where the responsibility is evenly divided. She says the lawyers are very concerned with the distribution of responsibility between the parties. You must have some sort of responsibility within each sprint. But stating who is responsible for what, is a great challenge, she states.

Transferring expertise between the customer and the supplier is also a challenge, she continues. The software development methods are a way to do effective expertise transfer and communication between supplier and customer. When you have an agreement, you use this to develop a requirement specification. But this is a bad way to transfer expertise. She says she attended the conference Smidig2007, and there a lecturer meant that the worst way to do transfer expertise is to write it down on a piece of paper. You forget that what you really are doing is transferring expertise when you deal with a requirement specification.

She states that there is nothing that obstruct agile methods, there are more myths. It is important to have a clear and concise distribution of responsibility. It is not very wise to write that *"we are together responsible for.."*. This is no good if you end up in a conflict. You have to write *"We share the responsibility 50 50"*. It is also important that the project manager has enough knowledge about the customer. She has seen many examples where the project manager lacks important knowledge about the client.

The SSA-S can be combined with competitive dialogue. Competitive dialogue is best suited if you for instance are going to buy an electronic patient record system, where there are a handful of suppliers. But this kind of system is very complex, and although you can use some standard components, a number of adjustments are required. What is great about competitive dialogue is that the customer get competence from the supplier, and they assist the customer to better understand his own needs. You first pre-qualify a suitable number of suppliers, and then you emit a needs assessment and an analysis of non-functional requirements. Then the suppliers are invited to a dialogue. During this dialogue, the suppliers create their own document, which describe their solution. As the dialogue continues, changes in the document are made, and it finally ends up as a tender document. The dialogue can be done in several phases, so you don't end up with too many suppliers at the end. It is also possible for the suppliers to get paid for their work during the process.

She thinks that when suppliers hand over offers, they tend to deliver over-dimensioned documents, which contain many things the customer did not ask for. But again, the customer does not know what they need. The requirement specification does not necessarily reflect their needs. Lawyers have trouble understanding this, she says. It is easy to just make a system according to the requirement specification, but then the customer does not get the system they really need. Many customers are also terrified of making mistakes, so they spend a lot of resources making an over-dimensioned requirement specification. Users are often included in the process after the requirement specification is finished. This was the case of a project she was involved in. The executives went on lengthy seminars with the supplier, and returned with opinions on how the system should work. The actual users were not included in the process until the contract was signed and the requirement specification was prepared. The users had other viewpoints than the executives on how the system should work, but the requirement specification was a real strait jacket. This caused the project to become real expensive because of all the changes that had to be made.

She states that it is important that the customer has not used up all their resources making the requirement specification. Also, they have to attend meetings with the supplier. A big problem is the lack of participants from the customer with decision-making authority in the project team. She believes that it is a significant difference between private and governmental customers when it comes to decision-making. A governmental customer has a strict hierarchy, and therefore it takes longer to reach a decision. She also thinks there is a general fear of decision-making in the Government. It is a real challenge to get customer representatives to participate in the project team and then getting decisions from them. Some customers just hire consultants to do the decision-making, but this will become a problem if the consultant does not know the client's company well enough. The consultants are at the risk of making decision which can be regarded as wrong, and the customer ends up with a system that is not properly rooted in real life.

She says she tries in the new SSA to get more presence from the customer, by having a number of partial acceptances of the solution. This forces the customer to make decisions. And if they later change their mind, this will be regarded as a change.

She likes how agile methods embrace changes. Jeff Sutherland said once: *"Changes for nothing, and nothing for free."* Most customers find out that they need a lot of changes, and this is very expensive in a regular contract. This because you get the change management process, and the whole bureaucracy that follows. In agile methods, if you decide you want a change, and the change is properly anchored in real life, it is implemented without any problems. She does not believe that it is possible to order exactly the system you want right from the start. It is a continuous process. It is also important to prioritize changes. You can't fill up the sprints with new features, without taking something out. The scope of the sprint must remain constant, she states.

It requires discipline to succeed with agile methods, i.e. SCRUM. She is afraid that many companies adapts their agile methods too much, and that this can have a negative effect on the result. If you work with an agile approach, you have to know what you are doing. For instance, she heard about a company that had an agile core in their development process. What do they mean by that, she asks?

She says that the customers have confidence in the work done by DIFI, and therefore they use the contract model. They trust the SSA. But SSA have a few challenges, and the perhaps biggest challenge is to get customer representatives. They have to be engaged and be able to make decisions. The customer believes they can buy the whole system, without participating. This often results in that they get a system they don't need, she explains

The SSA-S has flexible price mechanisms. It says nothing in the appendices about what price mechanisms you should use. This means you can have different agreements on each iteration. For instance, you may have fixed price on the first iteration, target price on the second, and time-and-materials on the third. It is also possible to choose if you want to approve each iteration, and this is very important to succeed with in practice. Else, you have to go through the change management process, and people react to this because it is so bureaucratic. So change management is very important, and you have to separate between changes that are inside the scope and changes that are outside the scope. If the changes are outside the scope, you must have a strict change regime, where you assess the consequences of the change.

Chapter 6

Analysis

In this chapter the statements from Chapter 5 are organized and arranged in different topics, in such a way that all statements concerning a topic is assembled. The chapter lays the groundwork for the discussion in Chapter 7, where the results are further analyzed and compared to findings from the literature study in Chapter 3.

The analysis was performed inspired by the techniques presented in the article *"A Purposeful Approach to the Constant Comparative Method in the Analysis of Qualitative Interviews"* written by Hennie Boeije [4].

6.1 Agile development

All of the interviewees have some sort of connection to agile methods, either they have experience working in agile projects, selling the agile concepts to customers or they adapt current or new contract models to suit agile methods.

6.1.1 Motivations for working agile

Reidar Sande and Marianne Selle agree that agile methods result in a product that the customer need and want. Mari Vestre agrees with them, and states that agile methods really work. She also states that agile development is a better way of transferring domain knowledge, than just using a requirement specification. Reidar also thinks agile methods make it easier to spot cost overruns. He states that you also get fewer problems with system- and integration tests, because the system has been thoroughly tested throughout the development process. Sverre has the opinion that collective responsibility is the best contribution agile has attributed.

6.1.2 Challenges with agile

Although all of the interviewees prefer agile methods, they also identify several challenges. Kjetil Strand and Jørgen Petersen says there exist little instructions on how to manage agile administratively, and they state that agility is not suited everywhere. Marianne says that the project team must have agile enthusiasts, agile does not work if the method is imposed the project team. Mari argues that you have to know what you are doing when running an agile project. She is also concerned that many adapt the agile method too much, in such a way that the methods lose much of its benefits. They all agree that agile do not work without the customer involvement.

6.2 Contracts and agile projects in general

Sverre Tinnen states that the best projects do not use the contract. It is then only used if a conflict occurs. Marianne agrees with Sverre. She considers the contract as a guidance, and says that it is mostly looked at if there is a disagreement. Also, she has observed that traditional ways of getting contracts with tender and bids are decreasing in number. More and more are using framework agreements. She also thinks that the increasing complexity in IT systems makes it harder to perform good cost estimates, which in turn makes it harder to set up a contract.

6.3 Agile methods and fixed-price contracts (SSA-U)

When talking about fixed-price contracts, all of the interviewees have experience with the SSA-U contractual model, which they perceive as a fixed-price contract model. Therefore, when talking about fixed price contracts in this section, we are therefore referring to SSA-U or similar contracts.

6.3.1 Motivations for using fixed price contracts

Marianne says that governmental customers use SSA-U by default, as if it was legally required. Mari supports this statement and says that the reason why customers use and trust SSA-U, is that they have faith in the work done by DIFI. Jørgen and Kjetil states that there are companies that have extensive experience with SSA-U, and therefore gain financially by being highly efficient in the start-up phase of the project, because of their good knowledge to the contract model.

Reidar argues that the reason why customers want fixed price contracts is to eliminate economical risk. Another reason to choose a fixed price contract is lack of trust between the customer and the supplier. And all the interviewees agree that trust is extremely important when working in agile projects.

6.3.2 Opinions on SSA-U

Marianne says that the SSA-U requires fixed price, fixed scope and a fixed time schedule. Reidar agree and states that he thinks SSA-U is a 100% waterfall oriented. Jørgen and Kjetil have a similar opinion; they state that the contract model is too narrow minded and waterfall oriented, and that it is very customer centric. In addition, they mean that the contract model lacks guidance on how to use it. On the other hand, Mari, the person responsible for SSA-U, argues that the contract model does not define which development method to use. She also adds that SSA-U is bound by governmental directions, which gives the contract model less latitude.

6.3.3 Challenges with fixed price contracts

Marianne says a problem with SSA-U is that it lacks good control mechanisms for requirement changes. The contract model does not take into account the time it takes to elucidate a change, and that this process can affect the already fixed delivery date. Sverre also thinks requirement changes are difficult when dealing with SSA-U.

Sverre says it is a challenge to use agile methods in a fixed price contract. Reidar agree and adds that there is much upfront work when having a fixed-price contract, and that this disagree with agile principles. Reidar also argue that you get less value for money when using a fixed price contract in an agile project. This is because you get an underlying conflict between the two parties where the customer wants most of his money and the supplier has a "least effort"-attitude. Marianne agrees with his perception and says it is one of the reasons she prefers framework agreements, since it helps avoid this war between the customer and supplier.

Sverre states that another challenge with fixed-price contracts in agile methods is finding an alternative to traditional documentation. The waterfall method often requires that each step of the process is extensively documented, which does not go well with agile principles. Sverre has experienced that using user stories as architectural design has worked well.

6.3.4 Positive aspects with fixed price and agile methods

Few of the interviewees are positive when it talking about using fixed-price contracts in an agile setting. However, Reidar has some good experiences regarding agile and fixed-price contracts. He notes that a prerequisite for succeeding in this setting is sufficient trust between the supplier and customer. He adds that a positive aspect of using agile methods with a fixed price contract is that as a supplier you are forced to create a simple and adequate solution. This means you avoid the temptation of making a *"fancy"* system with lots of features the customer did not ask for. Reidar has experienced just this in a project using a time & materials contract.

Sverre says that they have no problem working agile with a fixed price contract model, but he also states that they have used much time adapting their development methods to work well in combination with SSA-U.

6.3.5 Measures to be taken when working with fixed price contracts

Reidar, Marianne, and Sverre agree that it is important to have a constant scope when working with a fixed price contract. If the customer adds new features, other features with corresponding costs have to be removed from the scope. Mari also supports this statement.

6.3.6 Suppliers' desired improvements on SSA-U

Marianne and Reidar agree that something has to be done with the day fines in SSA-U. Day fines and agile development do not combine well. In general they mean that the contract model does not allow a full agile approach, and Marianne adds that they only use Scrum's reporting mechanisms when working with a project having a SSA-U contract.

6.4 Norwegian contractual standards

6.4.1 Experience with PS2000

Reidar explains that PS2000 can be interpreted as both a traditional and an agile contract depending on the readers view. Marianne on the other hand argues that PS2000 only promotes a lenient form of agile development. Jørgen and Kjetil explain that PS2000 has had a strong unofficial link to agile development, and that their effort to make agile guidelines to this contract will make the link official. They also argue that the guidelines will persuade agile skeptics to use agile methods, and agile fanatics to use PS2000 in their projects. The guidelines will also help unify the way the contract is interpreted. One of the ways Jørgen and Kjetil intend to do this is by replacing the original mapping of PS2000 to RUP with a mapping to Scrum.

6.4.2 Challenges with PS2000

The biggest challenge with using agile and PS2000 is the start-up phase, and the contractual foundation Jørgen and Kjetil says. Sverre agree, and argues that it is the detailed specification of the solution in startup-phase that makes PS2000 difficult to use in an agile setting. Reidar also agree with this, explaining how the upfront work hinders agile development. He adds that it is easier to combine it with agile methods given that a coarse description of the system is what you start with. Jørgen and Kjetil explain that PS2000's option to cancel after the first developing phase is of help if the customer lacks trust in the supplier. They also argue that the creation of a risk-matrix help divide the risk and might help to build trust between the developer and the supplier.

Another difficulty mentioned by both Marianne, Jørgen and Kjetil is reducing iteration size. Reidar explains that in his experience the length of an iteration is dependent on the customer, and the time it takes for them to reach decisions. Jørgen and Kjetil explain how they foresee problems with moving from longer to shorter iterations, and thereby having more frequent checkpoints. Marianne points out the same problem when she argues that even the current version has too much work associated with each check point. She also states that in her experience PS2000 is a little too formal, and that in many cases this could explain why many customers have a hard time understanding it. Most customers underrate the amount of customer involvement required she says. She also argues that the contract does not offer any good mechanisms for managing changes. Sverre partially agree with Marianne by stating that PS2000 does not stimulate real involvement more than any other contract.

Both Sverre and Marianne agree that when the customer outsources the project management, the need for formality increases. Marianne says that in her experience consultants hired by a customer has an inclination to choose PS2000. Reidar as mentioned earlier, points out yet another difficulty, namely the day-fines used to sanction the supplier for not delivering on the target date, this he argues is not compatible with the agile principles.

6.4.3 Development of SSA-S

The biggest challenge with creating SSA-S is to get the customer properly involved, Mari states. She says that she has tried to regulate the customer involvement contractually by including partial acceptance to each of the iterations. Sverre confirms this by saying that his reviews of the draft document has given him the impression that SSA-S focuses on distributed responsibility. Mari argues that in order to accomplish this she has mechanisms in the contract that allow the customer to accept or discard any iteration. Jørgen and Kjetil are however skeptical to this aspect of SSA-S and argue that there are few details on how the deliveries are tested to see if they are accepted, an aspect they themselves have struggled with when writing their guidelines. They go on to say that it seems like many of the ideas behind SSA-S are influenced by PS2000. Mari says that another way to get the customers involved is to introduce competitive dialogue, and allowing each iteration to use flexible pricing mechanisms. This will also help her achieve the goal of SSA-S; to get it as close to agile as possible.

6.4.4 Challenges with SSA-S

One of the limitations SSA-S has, according to Mari, is its binding to governmental directions. She says that while creating the contract, lawyers have been concerned with the legal implications of the proposed distribution of responsibility. She continues to say that lawyers have trouble understanding that the requirement specification does not necessarily reflect the system the customer need or want, which she claims is the basis for using agile methods. Jørgen and Kjetil on the other hand argue that SSA-S looks to be mostly based on input from lawyers, and say they are concerned that the engineers have not been consulted enough.

6.5 Change management

When it comes to change management Sverre states that prioritizing meeting summaries as documentation is needed. Reidar agree with Sverre. They argue that it is important to have documents to refer to if the customer question why the system is not in accordance with the original requirement specification. There are always changes from the original requirement specification, and Mari does not believe it is possible to order the exact system you need from the start. Therefore it is important to have some sort of system which documents the changes that have been made, in order to avoid conflicts.

Mari argues that changes are less expensive in agile projects, and that this might benefit the customer. However, it depends on how bureaucratic the change management process is. A strict and bureaucratic change management process conflicts with agile methods, according to Reidar. Mari says it is important to know whether the change requested is inside or outside the current scope. If it is outside you need stricter management with assessment of the consequences caused by the change. Reidar states that conflicts concerning whether a change is within the scope or not is common, and argues that such conflicts are natural in software development. He also states that the parties mostly reach an agreement on an early stage when it comes to such conflicts.

6.6 Customer

6.6.1 Getting customer onboard agile

Marianne, Sverre and Reidar agree that there can be some challenges when getting the customer onboard an agile project. Sverre also says that most of their customers trust their expertise in developing software and therefore leave this choice up to them. He continues to say that most customers want a sensible development method, and Reidar adds that many customers request agile because of its popularity. Marianne on the other hand says that in her experience not all customers dare use agile methods because of the increase in risk.

6.6.2 Customer involvement

Marianne, Reidar and Mari all agree that most customers do not know what they are getting themselves into. Reidar emphasizes that there is a need for customer education when it comes to agile development methods since few customers truly understand agile methods. Mari says that in her experience most customers think that they can order a system, and then get exactly what they want without more involvement. She also adds that most customers do not know what they want from the start. Jørgen and Kjetil argue that forced customer involvement is a supplier benefit, and lead to closer collaboration between the parties.

All interview participants agree that customer involvement is one of the keys to ensure a successful agile process. Reidar argues that agile methods do not work without the customer. Jørgen and Kjetil say that it is of crucial importance, but that it also contains a lot of pitfalls. Sverre explains that in his experience it is important to make friends with representatives within the customer organization in order to have them argue in favor of the final solution. He also adds that the customer representatives involved in the project often become more of a hostage than a participant, and that the representative often lacks the authority to make decisions necessary to ensure progress. Mari agrees that the customer representative often lack the authority, and adds that Governmental customers are faced with a strict hierarchy, which hinder swift decision making. In her experience customer representatives from Governmental customers also suffer from fear of making decisions, impeding the agile process even further. Jørgen and Kjetil explain that foreign customers often refuse PS2000s forced customer involvement, fearing that it will not be to their benefit. One of the project managers also state that communication between participants in the customer organization can cause problems, and adds that she has experienced this recently, when the person who made change requests did not consult the people in charge of the budget.

Mari explains that in her experience the customers are afraid of making mistakes and therefore spend much of their resources in making an over-dimensioned requirement specification to ensure they receive value for their money. She adds that the lack of customer involvement can be the result of using too much of the budgeted resources in specifying the product.

6.7 Trust

Reidar, Sverre and Marianne agree that the trust between a customer and the supplier can ease the development process immensely. Reidar argues that it is very important especially when working with agile methods. Marianne agrees and states that agile methods require trust in order to work. She continues to say that it is more important than the contract itself, and adds that in her experience lack of trust is the single biggest reason behind failed agile projects. Sverre says that one way to gain the customers trust is to persuade them that you have their best interest at heart. This he explains is most often done by maintaining a good relationship with the customer over time.

Jørgen and Kjetil explain that in their experience foreign customers are skeptical of revealing their own risks. Mari argues that a difficult aspect when developing contracts is to find a common ground where responsibility is divided evenly. She says that the goal is to find a solution that lies between fixed price and time & materials, and still is legally feasible. She continues to say that it is important to be able to share the responsibility and at the same time making it unambiguous.

6.8 Other experiences

While Reidar says he would choose PS2000 over SSA-U he also argues that in an agile setting he would prefer a contract that is based on time & materials. Marianne says she prefers framework agreements, since most of the risk then lies with the customer, who most often is also running the project. She argues that this also causes less conflict. Of the contractual standards that exist today she states that PS2000 is the closest match for agile projects.

Mari also points out that the process of choosing supplier leaves an abundance of documentation, both from the customer in the form of requirements, and the supplier in the form of specifications. This uses valuable resources from both parties and should be restricted. Jørgen and Kjetil agree with Mari and say that there is less risk in under-specifying than in over-specifying.

Chapter 7

Discussion

This chapter contains reflections regarding the results of the analysis and the literature. The sections in this chapter take a closer look at both the academia and industrial practitioners views on the different subjects.

7.1 General experiences with agile methods

7.1.1 Motivation

As mentioned in the Section 2.5, the biggest motivation behind using agile development methods is to make the development process as efficient as possible. Sverre Tinnen also argued that collective responsibility was the most important contribution made by agile development methods, and that involving the customer is a big step in the right direction. Mari Vestre commented that the close collaboration with the customer had another advantage. In her experience this meant that domain knowledge was transferred between the supplier and the customer much more efficiently than if it was done using the traditional requirement specifications.

Collective responsibility will also result in greater awareness within a team, and can combined with small iterations make it easier to manage a project of this sort. Both Sverre and one of the project managers, argued that it became easier to spot potential overruns when using agile development methods, which is in agreement with what is found in literature in Section 2.5.4.

The continuous communication between the customer and supplier is one of the biggest changes from plan-based to agile methods, and can therefore be difficult to achieve satisfactory. It does however also have its rewards. Marianne and Reidar agreed that one benefit was that the customer received the product they both needed and wanted. This can naturally be the result of any of the several agile principles or techniques, but it is most of all linked to the added customer involvement the methods requires.

When asked whether switching to agile led to any measurable benefits Sverre answered that to measure code quality or efficiency in projects before and after agile methods were used, could have revealed some of the differences, but that these would most likely depend on a myriad of variables beyond just the development method. Sverre also stated that the projects done using agile methods was *"perceived"* as more successful, which he argue might be just as valuable.

As mentioned in Section 2.5.5, added developer morale is often connected with working agile. This was confirmed by several of the interviewees. They reported that one of the major benefits were indeed developer motivation, and their overall positive response to this way of working. Marianne explained that just the aspect of moving yellow notes around on a white-board seemed to help motivate the developers. This is probably also a result of several different development techniques, but it can be one of those things that contribute in making the process slightly more efficient.

7.1.2 Challenges

Both Dybø and Dingsøyr, and Cohen et.al report in their reviews that both industry experience and research are leaning towards the use of agile in specific projects [8][10]. This is in agreement with statements from Kjetil Strand and Jørgen Petersen. They state that despite what agile proponents argue, there are settings in which other more traditional development methods are better suited.

Another challenge reported by Marianne Selle was that agile methods do not work optimally if the project team lacks one or more enthusiastic members. She had experienced several projects which had suffered failure because agile techniques had been imposed on the team. Marianne also reported that running a project required that the project manager had experience with agile methods. Similar arguments were presented in the research review by Dybø and Dingsøyr, namely that agile methods works best with experienced teams. This seems to indicate that the team members on the supplier side need to be properly educated in order for the methods to be beneficial, and that perhaps not all developers are as eager to use the methods as it is portrayed in success stories.

All of the interview subjects agree that agile methods are dependent on customer collaboration. It is however difficult to accomplish the level of collaboration that is necessary. The studies reviewed by Dybø and Dingsøyr do not report the same difficulties, but mention that having an on-site customer can become stressful on the development team. While it is common to argue in favor of adapting the development method according to the host organization, Mari is concerned that some adapt their methods too much. She says that in her opinion some of the companies has been to occupied with having their developers certified, or adding a buzzword to their resumé.

7.2 The buyer-supplier relationship

7.2.1 Motivation behind different contract models

Even though this thesis did not focus the research on the customers, the interviewees had opinions on what customers emphasized when they chose contract model.

Reidar Sande stated that a motivation for the customer to use fixed price contracts was to eliminate economical risk. This can be seen in connection with findings from the literature described in Chapter 3. Fowler argued that focusing on only the cost of the software is short sighted, because the software also has a business value [12]. Hence, if you do not get satisfactory software, you lose the business value. So one can argue that the customer only eliminate economical risk in the short-run when using fixed price contracts. In addition, Reidar stated that one get less value for money when using fixed price contract in an agile setting. This also indicates that customers that do not "save" money when using fixed price contracts.

SSA-U can be regarded as a fixed price contract according to the statements of the most all the interviewees, and based on the statements of Mari Vestre and Marianne Selle; governmental companies and departments use SSA-U by default. Mari stated that the customer trust the work done by DIFI which might be the reason behind the frequent use. It was also stated that the contract model was very customer oriented, and therefore popular with the customers. One also has to take into account that all the regulations governmental customer has to abide by are included. SSA-U is no longer the only development contract with this *"approval"*, but it most likely played a part in making it as popular as it is. Jørgen and Kjetil stated that many companies have extensive experience with

SSA-U, which makes it difficult to switch to another contract. The companies would then have to rebuild their expertise to fit a new contract model, which can be time consuming and a considerable financial expense. Besides it would take years to attain the same amount of experience which one benefited from using the first contract model.

Marianne stated that more innovative customers, who often hire consultants, are inclined to use PS2000. According to Jørgen and Kjetil, there is little migration between PS2000 and SSA-U. Both Reidar and Marianne prefers PS2000 over SSA-U when working agile, which could indicate that PS2000 is better suited for agile methods. It still looks like a majority of projects use SSA-U, which might be because either customer or supplier trust the contracts developed by DIFI, or has extensive experience with it. Another incentive for the customers to choose SSA-U over PS2000 can be because SSA-U to a large degree has the customers best interest at heart, while PS2000 might lean slightly more towards the supplier.

Lack of supplier trust is also pointed out by one of the interviewees as an incentive to use a fixed-price contract. This statement supports findings from the literature in Chapter 3. In Eckfeldt's et al. experience report the customer was of the opinion that in larger projects or when working with unknown consultants they preferred fixed-price contracts [5]. Poppendieck identified that lack of trust between the parties could result in a contract model that tries to prevent opportunistic behavior [16]. It was the interviewees' opinion that when the two parties trust each other, the contract is not very important. If there is not an adequate amount of trust in the buyer-supplier relationship, it is perhaps more likely that the buyer will prefer a fixed price contract model. If there exist enough trust in the relationship, the customer may agree to use other contract models which are more compatible with agile methods.

7.2.2 Customer issues

Customer involvement is crucial for succeeding when using agile development methods. The interviewees has pointed out some key issues when it comes to the customer's role in the project.

There are different experiences on how to get the customer on board agile development. It seems to be dependent on the buyer-supplier relationship. Ford, referring to his model of the buyer-supplier relationship, stated that it was impossible to have a true agile relationship in the first two stages of his model [9]. This was supported by Sverre's statements. But it is also important to note that it seems like the eagerness of the customer to use or not use agile methods might also depend on project properties such as size, criticality or known requirements, and will most likely be heavily influenced by previous positive or negative experiences.

Another issue is some customer's tendencies to make large and detailed requirement specifications. This was pointed out by several of the interviewees. Mari stated that she did not believe that anyone could specify their system down to the last detail, and do this correctly before the actual development started. Kjetil agreed and argued that such overspecifying would lead to a suboptimal process. This is also in agreement with what Wingård stated, and which is covered in Section 3.5.1 [26]. He stated that it was not possible too have a complete picture in advance on how a complex IT-solution should function, and argued it is better to use a coarse sketch as a basis.

Having an over-dimensioned requirement specification also indicates an old-fashioned software mentality. It is evident that some customers believe that they can buy custom-made software the same way they buy a computer or an office desk. Software development does however require communication, customer interaction and collaboration. It is also a continuous process which must be also taken into account in a contract model. Without an educated customer who understand these basics, it is hard to succeed using agile development methods.

7.3 Challenges with today's contract models

The interviewees identified several challenges with today's contract models. These are described in the following subsections.

7.3.1 Upfront work

In the literature, too much upfront work is identified as a challenge when using agile methods with fixed-price contracts. Reidar stated that having a fixed-price contract requires much upfront work and that it disagrees with the agile principles. This is largely commensurate with what Schah stated; that the estimation of time and cost is time consuming, thus reducing much of the benefits an agile approach would bring [19].

According to the statements of the interviewees, SSA-U can be regarded as a fixed-price contract. PS2000 has more flexible price mechanisms, and mainly uses the target-cost pricing model. Both Sverre, Reidar, Jørgen and Kjetil stated that it could easily become much upfront work in PS2000. Which gives a clear indication that it is not only fixed-price contracts that have trouble with upfront work, but that this also applies to other contract models.

7.3.2 Alternative documentation

The agile manifesto states: *"Working software over comprehensive documentation."* Traditional development methods often require extensive documentation throughout the project. This will also be reflected in traditional contract models, such as SSA-U which makes them difficult to use in agile projects.

Sverre explained that it was a challenge to find an alternative to traditional documentation when working with SSA-U. However, his experience was that using user stories as architectural design works. Although this is only one person's experiences, it is still very interesting. It shows that adaptions or tweaks can be made, in order to meet or bypass the contract requirement. It also indicates what other interviewees also stated, namely that SSA-U was created with plan-based development methods in mind. Both Marianne and Reidar had the opinion that SSA-U was a 100 % waterfall oriented.

7.3.3 Underlying conflict

Another challenge was identified when working with fixed-price contracts, such as SSA-U. Both Marianne and Reidar agreed that there is an underlying conflict between the supplier and the buyer when you are working under a fixed-price contract. This is supported by Poppendieck and Poppendieck who stated that a fixed-price contract is biased in favor at the customer at the expense of the supplier [15]. This makes it necessary for the suppliers to aggressively protect their interest, at the expense of the customer. This so-called *"war"* is another argument against fixed-price agile projects, and because of it neither Marianne or Reidar prefer fixed-price contracts when working agile. Marianne stated that when they have a project with a SSA-U contract, they cannot work truly agile, and that they in practice only use SCRUM's reporting mechanisms. Tursas states in his thesis that if the contract becomes too restrictive (as a fixed-price contract may be), it may hinder the effectiveness of the project, which is in accordance with Reidar and Marianne's statements [21].

This also apply to PS2000. It seems that it is possible to interpret the contract in either a traditional or an agile way. If it is interpreted as traditional, it can be very comprehensive.

7.3.4 PS2000: Iteration size and checkpoints

The length of a sprint in SCRUM is usually 2-4 weeks. After a sprint, a working partial solution is delivered. According to the developers of PS2000, adapting the contract model to SCRUM is a major challenge. PS2000 is originally mapped onto RUP, where the iterations are much longer. This also implies comprehensive checkpoints which was pointed out by Marianne. She stated that it was very difficult to do manage the checkpoint work on schedule after each iteration.

This is a clearly a challenge when using the contract, and something needs to be done to lighten or redistribute the workload related to each iteration. Jørgen and Kjetil stated that a solution to the problem is included in the new agile guidance to PS2000, but if this in fact solves the problem remains to be seen.

Based on comments made by Kjetil and Jørgen, using the iteration length of SCRUM, while having comprehensive acceptance tests after each iteration, will also be a challenge for SSA-S.

7.3.5 Formalism

There have been indications that today's contracts are to formal for them to be used along side of agile development.

Marianne stated that she thought PS2000 was a little too formal, without going into details. There are also indications of too much formalism in SSA-U, although this has only been stated indirectly by some of the interviewees.

Jørgen and Kjetil claimed that SSA-U, like every other contract developed by DIFI, is mostly based on the inputs by lawyers. Lawyers do not have the same understanding of software development as the engineers who work with it. Mari stated that lawyers, for instance, had a hard time understanding that the requirement specification does not necessarily reflect what the customer really needs. Too much input from the lawyers can result in a very formal contract. SSA-U is also bound by governmental directions which also might restrain the contract further.

However, contracts need formalism. Some of the interviewees stated that outsourcing of projects increase the need for formality. Too much formalism will result in an all-embracing contract. According to Tursas this will on one hand be very efficient for reducing uncertainties, but on the other hand it may hinder the effectiveness of the project [21]. Again, trust emerge as an important factor. If there is an adequate level of trust in the business relationship, there is no need for a very formal contract.

Both Sverre and Marianne agrees that the best projects do not turn to the contract unless a conflict occur. The suppliers use their money on making the customer happy by creating the system that they want and need. In these cases, the format and content of the contract would not influence the process at all, and one would be able to utilize whichever method, technique or tools that ensures the best possible end result. This could however only be achieved with adequate trust between the customer and supplier, and could result in trouble if a conflict occurs. If the supplier or customer has disregarded portions of the contract as a result of this goodwill, a conflict could escalate, and result in more severe consequences.

7.3.6 Change management

Change management is a key issue when it comes to agile contracts. *"Responding to change over following a plan,"* is the fourth item in the agile manifesto. Change management in an agile settings, can often become a problem. The customer is invited to contribute with frequent changes and reranking of priorities, which can easily cause delays. In a traditional plan-based development this would be handled through a formal and often bureaucratic process, where the the potential change first is delivered to the development team for estimation, a consequence analysis with the details regarding the change is then returned and the customer decided whether or not to issue a change request. If the request is delivered, the change is incorporated into the solution, and the appropriate documentation and requirements are updated along with any changes that might be required to other parts of the product. This is a lengthy process that do not abide the agile principles, it does however clarify the consequences in terms of cost and labor and document the actual request. Using it in an agile setting do however impede agility.

The problem that arises when using an agile approach, and therefore a less detailed specification of the product, is to decide whether or not a requested change is within the original scope or not. This is of particular interest if the contract is based on a fixed price, and the where the change causes considerable efforts on the part of the supplier. If the change is within scope, it should be handled without any further development cost or delays, but if it is not the contract needs to be revised. If changes are handled without any of the formalism used in traditional methods, this could lead to delays or added cost, without the awareness or consideration of the customer. This could lead to conflicts when the solution approaches completion, and it does not coincide with the original description.

It was pointed out by one of the interviewees that both SSA-U and PS2000 lacks agile control mechanisms for handling elucidation of requirement changes. The contract has to take into consideration the time it takes to elucidate a change, because this affects the project.

Sverre stated that their way of handling this challenge is to painstakingly take notes during any meeting where these kind of changes are requested. These notes together with any other details that might be required, are then made into summaries which are filed in some sort of meta system. The summaries can then be referred to if a conflict occurs, and will then document who made the request and whether or not this person was aware of the consequences. Reidar also argued the importance of documenting any changes, in order to have something to refer to when the system is not in accordance with the original requirements. Sverre stressed that these documents is seldom legally valid since there are few contracts to date that let you rank meetings summaries above requirements agreed upon in earlier states of the project. In a relationship where trust is established, the summaries, he says can help diffuse most conflicts. They do not, however guard you against conflicts that result in legal actions. There are possibilities of changing the ranking of requirements and meeting summaries, but neither of the subjects interviewed in this thesis had any experience with it.

Mari argued that changes in agile projects are less expensive, and therefore benefit the customers. She also said that cheap changes especially benefit customers who do not know exactly what they want, which in her experience is true for most customers. An experience related to this was also discovered in the interviews. One of the project managers reported that change management, in the agile sense, can easily cause delays, and increase in cost. She had experienced this, and the project had suffered severe delays, and massive budget overruns. She attributed this to the the customer representative, whom requested most of the changes, without regard for the economical consequences. The project manager, argues that it could have been avoided if the product owner was also the budget owner. This experience also underlines the need for good communication mechanisms, and shows that the challenge of transferring knowledge within or between organizations are still present even though the project is based on agile principles.

7.3.7 Customer involvement

Customer involvement is a critical success factor for agile development.

Continuous customer involvement is widely agreed to be one of the most important additions in agile development, and perhaps the most important necessity to ensure success. Reidar argued that without customer involvement it is difficult if not impossible to succeed in an agile project. Jørgen and Kjetil were emphasizing the same element when talking about the implementation of PS2000, saying that it is of critical importance to ensure an active customer. This particular contract also has the possibility to include a clause which forces the customer to participate actively. This clause, they said, are often excluded by customers who are apprehensive of what it actually entails, and the repercussion of a potential breach. They stated that it is their experience that foreign companies in particular are not comfortable with this added responsibility, and add that any contract where this clause is included would benefit the suppliers, and be of great help to ensure that the lack of customer involvement does not hinder the process. Sverre argued that this is probably the best way to go, but at the same time does not argue in favor of PS2000, stating that it does not stimulate real involvement more than other contracts.

Most of the interview subjects agreed that the customer mostly do not know what they are getting themselves into when agreeing to participate in an agile project. While forcing customers to participate through contractual clauses might ensure that they participate actively, their lack of knowledge will cause an unpleasant surprise if the resources that are needed have not been allocated on time, or at all. Marianne, Reidar and Mari agreed that the customers need to be educated in order to better prepare for what an agile project requires. Sverre said that in his experience there is also a risk that the product owner ends up as a hostage rather than a participant on the project team, this too might be an indication that the customer would have benefited from being educated, giving them information regarding their role. If the customer had better knowledge of the process, it might also reduce the amount of resources Mari argued most customers waste when specifying a system. It would according to Mari be quite the difference from what she often sees from customers today, where they expect to be able to define and order a system, then receive exactly what they want, without further involvement.

A benefit Reidar pointed out that also arise from close collaboration is that there will be someone on the inside of the customer organization that can defend the solutions decided upon by the project team. This might also help avoid possible conflicts if there are deviations from the original system description. While this is a benefit of having the customer participate in the process, both Mari and Reidar agreed that it can cause other problems as well. One of these is having participants without the authority to make decisions; this can like a bureaucratic change management system, cause delays and prevent the benefits agile would normally yield. Mari stated that this is often a problem when dealing with governmental customers, where the hierarchy of authority is strict, and decisions has to be approved near the top. She also added she has had experiences where participants from governmental customers had impaired decision making abilities. Having two examples of participation that failed, one could ask which would a supplier prefer; to have a participant without the authority to make decisions, and thereby potentially cause overruns not approved by the actual customer, or would it be having a representative that possibly cause delays by approving all changes bureaucratic. It would at least be overly optimistic to hope that people with decision-making authority will participate throughout an entire development process. This might however also become improved through education, spreading more information within the customer organisation about what is expected, could lead to the a midway between approving all changes at the top, and approving none.

7.3.8 Day fines

It appears that the interviewees do not think day fines and agile development go together. Such sanctions exist to protect the customer and apply pressure to the supplier in order to make them to fulfill their commitments. However, there are some issues regarding this. It was earlier pointed out that there were no good control mechanisms for elucidation of changes. The lack of such mechanisms means that the contract does not take under consideration the time it takes to elucidate a potential change. If there are many such change elucidations, it may delay the project. And this will result in day fines for the supplier, even though it is unfair. So without such control mechanisms, all the risk is with the supplier.

7.4 Trust, a key issue for agile contracts

All of the industry experts agreed that trust between the supplier and customer is very important. This has also been identified earlier in this chapter. Most of the difficulties discussed in this chapter would not have been an issue at all, if the participants had faith that the other party would work towards the common good. Trust is especially important when working with agile methods because of the uncertainty the parties are faced with when starting a new project. Agile methods is also argued by many throughout this discussion to work best if the risk, and responsibility is more evenly shared between the participants. One way to do this is to shift the payment-method from fixed-price to time & materials or target-price. This will under normal circumstances mean that the supplier who seemingly took little risk before, now has to get more involved. If trust is already established this leap of faith would be easier.

Both Marianne and Reidar agreed that agile methods require more trust than traditional methods. Marianne went as far as saying that trust is actually more important than the contract itself, adding that in her experience trust is the most common reason behind failed agile projects. There is no easy way to achieve trust however, Sverre stated that in order to gain the customers trust you have to maintain a good relationship over time, until the customer knows that you have their best interest at heart.

Chapter 8

Conclusion

This chapter concludes this thesis, and present proposals for further work.

8.1 Conclusion

The research goal of this thesis was to discover industry experiences regarding the use of agile development methods in projects based on a contract designed for plan-based development methods. The research was further divided into two research questions, the first being what experiences the industry has with agile development methods in general, and how the choice of contractual model affect their ability to use these methods. Suppliers has positive experience using agile, and several benefits were identified, where the most important includes increased business value, perceived increase in productivity and heightened morale among the project team members. It is indicated that one of the biggest challenges with agile development methods in general, is lack of customer knowledge, which can directly or indirectly be the source of other experienced difficulties.

There were also identified several challenges with today's contract models. The most important challenges were customer involvement, much upfront work, and bureaucracy. These challenges causes the suppliers to make adaptations in their agile development process. The challenges also reduce some of the benefits with agile development. This especially applies to contracts meant for plan-based development methods. Although contract models to some extent hinder agile development, suppliers tries to find ways to maximize the benefits of using the methods. This is done by putting the contract aside. Suppliers focus on pleasing the customer, and as long as the customer is confident that their needs are going to be met, the contract is put aside.

The second part of the research goal was to identify the work done in order to adapt existing contractual standards towards agile methods. The interviews and literature shows that because of the challenges identified with contracts and agile development, the users of contractual frameworks have requested more updated contract models that facilitate agile development. There are currently two ongoing projects in Norway with this intention. PS2000 was to be revised, but it was discovered that the contract framework already complied with SCRUM. Therefore, a guidance on how to use PS2000 with SCRUM is now being developed, and is planned to be released soon. The other project is the development of SSA-S. It is based on SSA-U, but is thoroughly revised in order to comply with agile development. A draft is currently out for hearing, and the contract is due out this summer.

8.2 Further work

There has been done very little research on the topic of how agile development is used under different contract models. At the same time the industry displays a large interest in the subject which the results of this thesis clearly shows is a challenge today. In order to use the results from this and to further the research on the subject, the following items are suggested for future work.

One approach is to do case studies of projects using agile development and identify challenges with the contract model used. This would results in first-hand knowledge of the potential issues, and confirmdeny the challenges and experiences identified in this thesis.

Another interesting approach is to collect experiences from the industry regarding the soon to be released versions of SSA-S and the agile guidance to PS2000. The research could investigate if the issues identified in this thesis were improved or completely solved. This would however have to wait until after the industry have gained an adequate amount of experience with these new contracts.

The customer's viewpoints are not represented in this thesis. Research focusing on their perspective might therefore broaden the understanding of issues like customer collaboration, the distribution of risk and the actual improvements caused by the use of agile development methods.

Acronyms

ACM Association for Computing Machinery

IEEE Institute of Electrical and Electronics Engineers

SSA-U The Norwegian Governmental Standard Agreement for Software Development (standard)

SSA-S The Norwegian Governmental Standard Agreement for System Development (agile version)

SSA The Norwegian Governmental Standard Agreements

EVISOFT EVidence based Improvment of SOFTware engineering.

SINTEF The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology (NTH).

NTNU The Norwegian University of Science and Technology.

UiO The University of Oslo.

ICT Information and Communication Technology.

NCS The Norwegian Computer Society

PS2000 Project Management 2000

RUP Rational Unified Process

SOA Service Oriented Architecture

ROI Return On Investement

DIFI The Agency for Public Management and eGovernment

XP eXtreme Programming

DSDM Dynamic Systems Development Method

FDD	Feature Driven Development
TDD	Test Driven Development
IT	Information Technology

Bibliography

[1] Ps2000 kontraktsstandard for leveranse av programvare m. m. Technical report, The Norwegian Computer Society, 2001.

[2] Scott Ambler. Survey says: Agile works in practice. `http://www.ddj.com/architect/191800169?cid=Ambysoft`, 2006. Visited 25.05.08.

[3] Robert Biddle Angela Martin and James Noble. When xp met outsourcing. 2004.

[4] HENNIE BOEIJE. A purposeful approach to the constant comparative method in the analysis of qualitative interviews. *Quality & Quantity*, 36:391 – 409, 2002.

[5] Rex Madden Bruce Eckfeldt and John Horowit. Selling agile: Target-cost contracts. 2005.

[6] Robert N. Charette. Why software fails. *IEEE Spectrum*, September 2005.

[7] Martin Höst Magnus C. Ohlsson Björn Regnell Anders Wesslén Claes Wohlin, Per Runeson. *Experimentation In Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.

[8] Mikael Lindvall David Cohen and Patricia Costa. An introduction to agile methods. *Elsevier*, 62, 2004.

[9] Håkan Håkansson David Ford, Lars-Erik Gadde and Ivan Snehota. *Managing Business Relationships*. John Wiley & Sons, 2003.

[10] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, January 2008.

[11] Jim Highsmith et.al. History: The agile manifesto. `http://agilemanifesto.org/history.html`, 2001. Visited 25.09.07.

[12] Martin Fowler. Martin fowler. http://www.martinfowler.com. Visited 24. Feb 2008.

[13] Tom Gilb. *Principles of Software Engineering Management*. Addison-Wesley Professional, 1988.

[14] Barbara Kitchenham. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University, 2007.

[15] Tom Poppendieck Mary Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.

[16] Tom Poppendieck Mary Poppendieck. *Implementing Lean Software Development: From Concept To Cash*. Addison-Wesley, 2007.

[17] Pete McBreen. *Questioning Extreme Programming*. Pearson Education, 2003.

[18] Jussi Ronkainen Pekka Abrahamsson, Outi Salo and Juhani Warsta. Agile software development methods. review and analysis. *VTT Publications*, 478, 2002.

[19] Peter Schuh. *Integrating agile development in the real world*. Addison-Wesley Professional, 2005.

[20] George Mangalaraj Sridhar Nerur, RadhaKanta Mahapatra. Challenges of migrating to agile methodologies. *Communications of ACM*, 48:72 – 78, 2005.

[21] Samuli Tursas. A relationship-oriented viewpoint on agile software contracting: a multiple case study. Master's thesis, University of Oulu, 2007.

[22] Dapzury Valenzuela and Pallavi Shrivastava. Interview as a method for qualitative research. `http://www.public.asu.edu/~kroel/www500/Interview%20Fri.pdf`, 2007. Visited 23.11.07.

[23] Mari Vestre. Ny ssa underveis: systemutviklingsavtale med vekt på smidige (agile) metoder og konkurransepreget dialog. Technical report, The Agency for Public Management and eGovernment (DIFI), 2008.

[24] Survey: The state of agile development. `http://www.versionone.com/pdf/StateofAgileDevelopmentSurvey.pdf`, 2007. Visited 25.05.08.

[25] Juhani Warsta. *Contracting in software business*. PhD thesis, University of Oulu, 2001.

[26] Trond Wingård. Fastprisdilemmaet - også kundens dilemma. *Computerworld*, 2003.

Appendix A

Search approach

The goal of this literature search will be to get an overview of the state of the practice. This will make it clear where the gaps in the research is located, and at the same time give insight into documented results.Kitchenham emphasize that the research question is the most important part of any systematic review. The research question used by us at this point is temporary since the review itself will most likely help us to further define it. The following research question is used when defining the search strategy:

> What are the benefits and difficulties in using various contract models in agile development projects?

A.1 Search strategy

After defining our research question we followed Kitchenhams recommendation, and made it the basis of our search strategy. After some iterative search testing we broke down our research question into their most important pieces, agile and contract, and found synonyms, abbreviations and alternative spelling for these. We also decided to add "agile case study" just in case our initial searches did not yield any good results. These strings were then combined in different ways and results recorded.

The table A.1 displays the strings we decided that represents most kinds of agile development. It also contains the strings we decided would catch the most used contract types.

Agile	Contract
agile, lean, incremental, iterative	contract, target price/cost, fixed price/cost, time and materials/cost/price, price, cost, target schedule/time, shared benefit, case study.

Table A.1: Search strings

Our search will span over some of the most widely used search engines on the Internet. This also included a search within the network of Norwegian libraries. Table A.2 contains an overview over the search-engines used.

Name	Description
Google Scholar	Google's search engine for scholarly literature. It contains articles, books, theses from several sources.
Web of Science	A search engine that uses among others the "Science Citation Index Expanded" database, which indexes many of the most know journals in several sciences, including computer science.
ACM Digital library	The Association for Computing Machinery (ACM) digital library is a vast collection of citations and full text from ACM journal and newsletter articles and conference proceedings.
IEEE Xplore	Institute of Electrical and Electronics Engineers (IEEE) Xplore is a digital library providing access to the technical literature in electrical engineering, computer science, and electronics.
BIBSYS	BIBSYS supplies library systems to approx. 100 libraries and institutions of higher education in Norway, such as the university libraries, the National library, college libraries, and a number of research libraries and institutions.

Table A.2: Overview of the available search engines

A.2 Search documentation

Google Scholar

Search string	**Hits**	**Relevant**
agile contracts	about 6000	7

Table A.3: Google Scholar search for *agile contracts*

Search string	**Hits**	**Relevant**
agile fixed-price	162	0

Table A.4: Google Scholar search for *agile fixed-price*

Search string	**Hits**	**Relevant**
agile target-price	21	0

Table A.5: Google Scholar search for *agile target-price*

Search string	**Hits**	**Relevant**
agile case study	35	3

Table A.6: Google Scholar search for *agile case study*. Search string had to be in title.

Web of Science

Search string	**Hits**	**Relevant**
agile contracts	8	0

Table A.7: Web of Science search for *agile contracts*

Search string	**Hits**	**Relevant**
agile case study	81	3

Table A.8: Web of Science search for *agile case study*

ACM Digital Library

Search string	**Hits**	**Relevant**
agile contracts	303	0

Table A.9: ACM Digital Library search for *agile contracts*

Search string	**Hits**	**Relevant**
agile case study	919	2

Table A.10: ACM Digital Library search for *agile case study.* Went through the title of the first 120 hits.

Search string	**Hits**	**Relevant**
agile fixed-price	6	0

Table A.11: ACM Digital Library search for *agile fixed-price.*

Search string	**Hits**	**Relevant**
agile target-price	0	0

Table A.12: ACM Digital Library search for *agile target-price.*

IEEE Explore

Search string	**Hits**	**Relevant**
agile contracts	29	3

Table A.13: IEEE Explore search for *agile contracts*

Search string	**Hits**	**Relevant**
agile case study	86	2

Table A.14: IEEE Explore search for *agile case study*

Search string	**Hits**	**Relevant**
agile fixed-price	2	2

Table A.15: IEEE Explore search for *agile fixed-price*. The two relevant articles were already selected from a search in another database.

Search string	**Hits**	**Relevant**
agile target-price	0	0

Table A.16: IEEE Explore search for *agile target-price.*

Appendix B

Interview Guides

All the interviews start with an introduction of the interviewers and the purpose of the interview. This will lead to a short presentation of our work and what we are interested in. We will also give a short explanation of how the results of the interview will be analyzed and presented. Finally the interviewee is asked whether or not the interview can be recorded.

B.1 Head of the software department

Details
What is you role in software development projects?
How much experience do you have with software development?

Experiences
Do you have any preferences when it comes to development method?
What experience do you have with agile development methods?
What are your experiences with customer involvement?

Contract
What positive and negative experiences do you have with contract models and agile development?

What sort of contract model do you prefer (independent of the cus-

tomer′s point of views)?
To what extent does the choice of contract model affect the customer interaction?
How are requirement changes handled?

Is it hard to convince the customer of a time-and-materials contract?
What type of customer (i.e. governmental, private sector) goes for a time-and-materials contract? Why do they choose this contract?
What sort of experience do you have with fixed-price projects and agile methods?
Have you experienced any examples on the customer regretting having a fixed-price contract? (refer to Fowler′s fixed-price mirage)
Have you any experience with the PS2000 contract model or SSA-S?

B.2 Project manager

Details
What is your role in the development project?
How much experience do you have with software development?
What is your preferred development method?
What kind of experiences do you have with agile software development projects?
What is your experiences with on-site customer involvement?

Project
To what extent do the customer participate in the project?
How does selection of development method happen?
How are requirement changes handled?

Contract
What sort of contract model is used in the current project?
What is your preferred contract model? Why?
How do you perceive that the contractual model affect the development cycle?

B.3 Developers

Details
What is your role in development projects?
Do you have any preferences when it comes to development methods?
How long experience do you have with software development?

Experience
What experiences do you have with agile development methods?
Do you have any experience with on-site customer involvement? What are these?
How do you perceive the the process of changing requirements and the negotiation of these?

Contract
Are you aware of the kind of contractual model used in projects you participate in?
Do you notice any effects from different contractual models?
Do you have any preferences when it comes to contractual model?

B.4 Customer

Details
What is your main business area?
How frequent do your company make use of external software developers?
What kind of technology and domain does these projects cover?
How do you manage these projects?

Experiences
What kind of experience do you have with software development projects?
What kind of experience do you have with projects using agile development methods?
Have your company any experience with active on-site participants

in development projects?
What experiences do you have with changing requirements and the negotiation of these?

Contract
What kind of experience do you have with different contractual models (fixed-price, target-price, time&materials, other)
Do you have any positive or negative experiences with any particular contract model?
What do you consider important when choosing contractual model?
Do you have any positive or negative experiences with risk management?
What contractual model do your company prefer, and why?
Do you have any positive or negative experiences when handling requirement changes?
What do you consider prerequisite for choosing a flexible, high-risk contractual model?

B.5 Contractual expert SSA

In general What kind of problems need to be resolved?
What is required in order to ensure that the new model gets used?
Is it possible to use agile methods as they are described or do they have to be adapted?
What kind of potential do agile methods have?
What are the biggest challenges of plan-based projects?
Who are usually in charge of changes that needs to be addressed?
What needs to change in order to get the most benefit from agile methods?

SSA-S Why are you developing a new version of The Norwegian Governmental Standard Agreements (SSA)?
What is competitative dialogue?
What is required of the supplier when using competitative dialogue?
What does it mean to have flexible pricing mechanisms?
What are the differences from SSA-S to SSA-U?
Which parts of SSA-S has been difficult to make agile?

What do you think will be the biggest challenges when one start using SSA-S?

B.6 Contractual expert PS2000

In general What kind of problems need to be resolved?
What is required in order to ensure that the new model gets used?
Is it possible to use agile methods as they are described or do they have to be adapted?
What kind of potential do agile methods have?
What are the biggest challenges of plan-based projects?
Who are usually in charge of changes that needs to be addressed?
What needs to change in order to get the most benefit from agile methods?

PS2000 Why are you developing a guide to PS2000?
What are the challenges do you foresee regarding PS2000 and agile development methods?
What kind of customers prefer PS2000?
How do the start-up phase of PS2000 suit agile development, are there any challenges?

B.7 Industrial experience

How do one persuade a customer to get onboard an agile project?
How is uncertainty and risk dealt with?
Do agile projects present any new requirements from the customer?
How do you regulate the customer's participation and responsibilities?
Do agile methods make it easier to deliver good solutions?
What changes are needed in the current contractual standards and/or regulations?
What is your preferred model contract?
Have you any experience with agile methods in a fixed-price project?
Do you have any experience with Norwegian contractual standards?

(PS2000?, SSA?)