Master Thesis
Software Engineering
Thesis no: MSE-2003-12
2003 June

A comparison of lifecycles

- Agile software processes vs. projects in non-Agile software companies

Stefan Saarnak and Björn Gustafsson

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Contact Information:
Authors:
Stefan Saarnak
E-mail: zarnak@tiscali.se

Björn Gustafsson
E-mail: bjorn@eventus.se

University advisor:
Yvonne Dittrich
Department of Software Engineering and Computer Science
yvonne.dittrich@bth.se

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.ipd.bth.se/
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

In the software industry a number of different software processes has been used throughout the years to address known problems with software development, despite their intention complains has been raised that some of these are too bureaucratic. The Agile Alliance was formed in 2001 and aimed to solve this problem, they developed a manifesto and twelve principles which are supported by all Agile software processes. The purpose with the manifesto and its principles is to uncover better ways of developing software and these are by many intercessors of Agile seen as common sense and not completely new ideas.

The aim with this master thesis is to answer the question if companies that explicitly claim that they do not use any Agile software process are already applying some of these ideas since they are thought of as obvious and common sense. The comparison in this thesis is performed between the project lifecycles used in specific projects by five non-Agile software companies and four identified lifecycle characteristics and two more general characteristics of the Agile software processes Extreme Programming (XP) and Dynamic Systems Development Method (DSDM). The result from the analysis of these interviews has shown that it is very difficult to decide if a software company really is working as described by XP or DSDM, this is due to that many different factors affect the final outcome. For example type of project and is the software company using different software processes for different kinds of projects. Since we just covered specific projects we were only able to conclude with absolute certainty actions that really were performed in just these projects. The project lifecycles of these software companies had some similarities with the above mentioned Agile software processes, but as a whole the analysis showed that they are quite different due to that two very important characteristics according to us, namely iterative development and frequent releases, were not applied by any of the software companies and that their project phases differed tremendously compared to XP and DSDM. Our common sense hypothesis for Agile software development was shown in this investigation to be incorrect since important activities were not performed by any of the software companies. Instead of using an iterative approach with frequent releases they all followed sequential waterfall like software processes.

Keywords: Agile software development, Software process lifecycle, XP, DSDM

ACKNOWLEDGEMENTS

We would like to thank everyone that has contributed to this master thesis, especially:

- The five Project managers who participated in the interviews and gave us feedback on the interview summaries and analysis.

- Our advisors Yvonne Dittrich and Christina Hansson, Blekinge Institute of Technology (BTH), for the support, guidance and constructive criticism.

CONTENTS

1 INTRODUCTION

In this first chapter we present the background to why we chose this topic and slant for our master thesis. A definition of the term software process is also presented in this chapter to clarify what we really mean with this term, this is done since we frequently are discussing different software processes and their lifecycles throughout this master thesis.

1.1 Background

The idea for this master thesis emerged during the fall of 2002 when we participated in and accomplished two courses at BTH in Ronneby, Sweden. These courses were ‘Project and Quality Management’ and ‘Advanced Software Topic – Software Processes’, in the first one we e.g. performed an interesting teamwork where we scrutinized a software process and in the second, we looked deeper into techniques and lifecycles used in different software processes.

During these two courses we also discussed Agile software development as one of the new methodologies to develop software. It was after these discussions we started to get interested in diving even deeper into Agile software development and its different methods. Another event that inspired us were two speeches held by the creators of the Unified Modeling Language(UML) Grady Booch and Ivar Jacobson in Stockholm during the fall at a Rational conference that we attended, where they discussed the future and their beliefs in how Agile software development will grow strong in the software development society.

We then examined the Agile manifesto and its twelve principles that Agile software development is based upon, and found that a number of the issues discussed were very obvious to us and felt like common sense. So our first thought was that even software companies which not explicitly claim that they are using an Agile software process must follow a majority of these guidelines. With this insight in mind and our knowledge in software processes we formed our master thesis proposal, which focus on comparing project lifecycles of non-Agile software companies to the lifecycles of Agile software processes. The aim with this master thesis is to perform a comparison between the lifecycles used by five non-Agile software companies and the ones used in the Agile software processes XP and DSDM.

1.2 Software process

The term software process in this master thesis refers to the activities associated with software development. The steps and procedures performed by software companies during development of a software product can follow either a self defined software process or a more universally known one. There exists a great deal of different software processes used by software companies and organizations for software development. In this master thesis we will present two well known Agile software processes XP and DSDM, but also the software processes used by five different software companies which we will compare to the Agile ones.

In the software community a debate about prescriptive versus descriptive software processes have been going on for a couple of years. The more traditional software processes advocates a prescriptive approach where the software process steps are defined early and in detail, and the goals for the project are fairly stable during the entire project. On the other side is the relatively new Agile concept which advocates a descriptive approach where software process steps and goals are determined dynamically. These decisions are based on experiences from previous software process steps, similar experiences gained prior the project, and on requirements and environment changes during the development (Turk et al. 2002). Our starting point concerning this subject is presented in section 2.2 Problem description.

Sommerville (1982: pp. 8) has the following general definition of a software process: "A software process is a set of activities and associated results which produce a software product." These activities can vary from one software process to another, but four of these are fundamental and used by all software processes, these are:

1. *Software specification*, the software functionality and its operation constraints must be defined.

2. *Software development*, to meet the specification the software must be produced.

3. *Software validation*, to ensure that the software does what the customer wants it must be validated.

4. *Software evolution*, to be able to meet the customers changing needs the software must evolve. (Sommerville 1982)

2 SCOPE AND STRUCTURE

2.1 Introduction

To be able to follow the content of this master thesis and understand how the work will be conducted a description of why and on which ground it is made will be necessary. The content is limited by a description of the problem area and to avoid unnecessary dissipations some limitations and assumptions are made.

A target group and four research goals are made to make it even easier to understand the purpose of this master thesis. The road to reach these goals is described in Figure 2-7 and it will show how the work will be structured.

2.2 Problem description

A vast number of different software processes have been used throughout the years to address the problems with software development. One of the main purposes with these software processes is to structure the way of working by emphasis on planning, and thereby help the development team to succeed with their project. Another purpose is to make software development more predictable and more efficient (Fowler 2000). Examples of these software processes are Rational Unified Process (RUP) and the Software Engineering Institute’s Capability Maturity Model (CMM) (Charette 2001). Despite their intention criticism has been presented that they are bureaucratic (Fowler 2000).

An alternative methodology to the software processes mentioned above was developed in February 2001 when the Agile Alliance was formed (Fowler 2002). A manifesto was defined for encouraging improved ways of developing software, based on this manifesto they also agreed upon a collection of principles that defines the criteria for Agile software processes. Examples of Agile software processes that support the manifesto and its principles are XP, Crystal methodologies (Crystal), SCRUM Software Development Process (SCRUM), and DSDM (Cockburn 2002).

The starting point of this master thesis is that much of the Agile manifesto and its principles are already used in the industry. Hence, even non-Agile software companies must follow a majority of these guidelines. This will be investigated by conducting five interviews with software companies that explicitly claim that they do not use any Agile software process. The focus of the interviews is to find out how these software companies are performing their software development. Our starting point concerning the software process used by these software companies is two folded. We believe that software companies in general have a prescriptive software process written down that they should follow if possible or no software process at all. Despite this, we believe that a more descriptive approach is used during the development to cope with unexpected events e.g. changing requirements, failures during development. The result from the interviews will be used for a comparison between the project lifecycles used by these software companies and the lifecycles of the Agile software processes XP and DSDM. The reason for choosing these two Agile software processes were two folded, first of all we felt that these two were the most interesting ones and secondly Charette (2001) presents that these two are some of the most used Agile software processes.

2.3 Target groups

Any skilled software engineer working with software processes or participates in projects where any software process is used.

Any students studying within Software Engineering or Computer Science whose future will involve participating in software projects, development of software processes, or software process improvement.

Any student studying within Software Engineering or Computer Science or skilled software engineer wanting to know more about Agile and its manifesto and principles, the Agile software processes XP and DSDM, or how five non-Agile software companies performing their software development.

2.4 Research goals

1. Investigate and define the manifesto and principles in Agile to clarify the base for Agile software development.

2. Define the characteristics for the lifecycles of the Agile software processes XP and DSDM.

3. Conduct qualitative interviews to find out what kind of project lifecycles that are actually used in practice, not the defined one, by five software companies that explicitly claim that they do not use any Agile software processes.

4. The result from the interviews will be used for a comparison between the project lifecycles used by these software companies and the lifecycles of the Agile software processes XP and DSDM. The aim with the comparison is to see if any of the five software companies are using some parts of the already mentioned Agile software processes lifecycles.

2.5 Limitations

To be able to fit the work within the time limits of this master thesis the survey is limited to five software companies with one interviewed person per company.

The five persons that will be interview are project managers, which may lead to a distortion of how certain areas were performed compared to how the projects actually were performed. E.g. who made the decisions the customer or the project manager?

2.6 Assumptions

This master thesis will not deal with the question if the use of software processes is profitable. A software process must be considered needed otherwise the reader will probably not gain anything from reading this master thesis.

The software companies that we are going to interview do not use any Agile software process since this master thesis focuses on to see weather these software companies maybe already are using some parts of an Agile software process lifecycle.

2.7 Structure

The research in this maser thesis will follow the roadmap that is shown in Figure 2-7. As the figure shows, we are going to use a combination of two research approaches, interviews and a literature survey. These two approaches have been selected to ensure that our research goals will be reached.

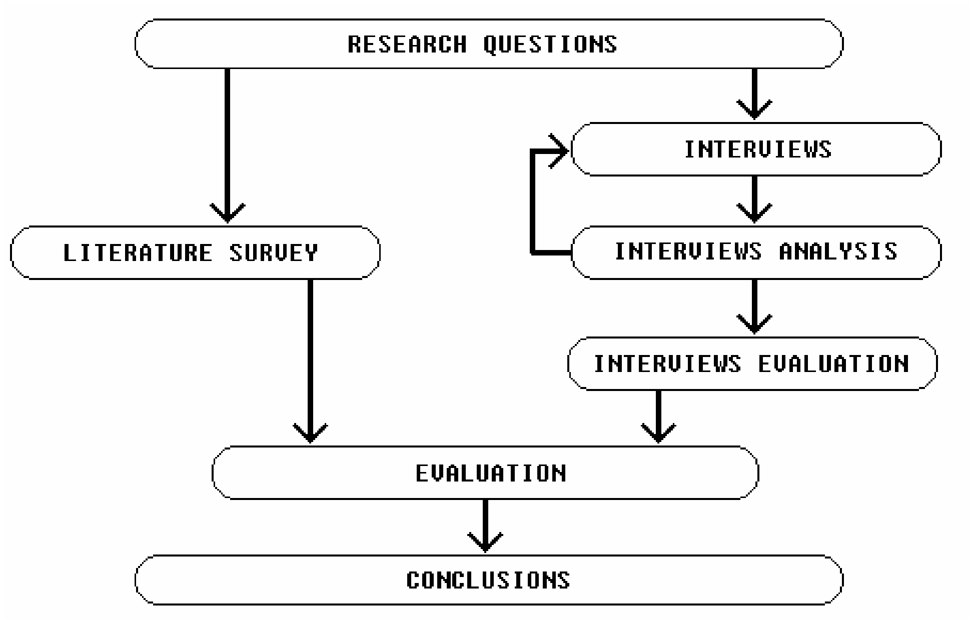

Figure 2-7. A description of how the work of this master thesis will be conducted.

During the **literature survey** we will gather information and by this gain knowledge in different areas related to our topic. These findings will be presented in this report as follows, software process lifecycle model in chapter 3 “Software process lifecycle model”, Agile software development in chapter 4 “Agile software development”, and interview techniques in chapter 5 “Investigation approach”. The aim with the literature survey is to reach the first two of our research goals.

During the **interview process** five software companies will be interviewed, this will accomplish our third research goal. The result will be presented in chapter 6 “The survey”. As can be seen in Figure 2-7, we will if necessary return to the software companies after analyzing the results from the interviews with complementing questions.

When both the literature survey and interviews are carried out, we will perform an **evaluation**. This will be done to evaluate how the results from these two can be related to each other. The aim with the evaluation is to accomplish our fourth and final research goal, this will be presented in chapter 7 “Analysis of interviews”.

The goal for the **conclusion** part is to summarize the results from the literature survey and interviews to be able to present our conclusions about the initial research questions, this will be presented in chapter 8 “Discussion and conclusions”.

3 SOFTWARE PROCESS LIFECYCLE MODEL

3.1 Introduction

In this chapter, the software process lifecycle model also known as just the software process model is defined and examples of two of these models are also presented to clarify and visualize how these can be structured. The two models covered are the waterfall model and the spiral model, the reason for choosing these two are that they are well known and recognized by the software developing society as two important models. The original lifecycle model was defined by Winston Royce in 1970 in the publication of the paper "Managing the Development of large Software Systems: Concepts and Techniques" (Royce 1998), this is now known as the classic waterfall model (Davis et al. 1988). The spiral model was introduced in 1988 by Barry Boehm and it is based on experience with various refinements of the waterfall model during several years (Boehm 1988).

The notion of a software lifecycle or software process emerged during the 60´s after a number of important software projects failed (Sommerville 1996). These problems with software development lead to the creation of the waterfall model and later also other models with the aim of reducing similar failures.

3.2 Definition

"The primary functions of a software process model are to determine the *order of stages* involved in software development and evolution and to establish the *transition criteria* for progressing from one stage to the next. These include completion criteria for the current stage plus choice criteria and entrance criteria for the next stage." (Boehm 1988: pp. 61)

The lifecycle model is just an abstract representation of a software process and it only provides partial information from a particular perspective about the software process it represents (Sommerville 1982). In other words the lifecycle model can be seen as a list of things to do since it in a step by step fashion breaks down the software process. A software process can use one or more lifecycle models, this depends on how the software process is defined. An example of a software process that uses two lifecycle models is the Microsoft Solution Foundation (MSF) where the waterfall- and the spiral model are combined (Microsoft Corporation 1999).

3.3 Waterfall model

The waterfall model emphasizes on development through sequential phases with predefined documents, milestones, and reviews after each phase, thus a prescriptive model. The result of each phase is normally one or more documents which should be approved before the next phase begins. Jacobson et al. (1999: pp. 450) presents a definition of the basic ideas behind the waterfall model that clearly explains the main characteristics of this model, "A system development approach in which the development is arranged as a linear sequence of work in, for instance, the following order: requirements capture, analysis, design, implementation, and test." In practice thus these phases overlap and information is passed between them. This leads to that the waterfall model is not a true linear model, instead it involves sequences of small iterations of development activities (Sommerville 1982).

In Figure 3-3 the different phases of the waterfall model is presented and here it is easy to see with results cascading from one phase to another why the model is called the waterfall model. The main stages of the waterfall model can easily be related to fundamental development activities that are used for software development. Here follows a description of these activities:

1. *Requirements analysis and definition*, the constraints, services, and goals of the system are determined through discussions with the users of the system. These then serve as a specification for the system being developed.

2. *System and software design*, in the system design process all requirements are partitioned, they fall into either hardware or software systems. This helps to identify the overall system architecture. The software design then includes finding and describing fundamental abstractions of the software system and relations between them.

3. *Implementation and unit testing*, the software design from the previous step is used to implementing the system as program units or a set of programs. The unit testing is verifying that the agreed upon specification is followed as planned.

4. *Integration and system testing*, all parts of the system are integrated into an entire system and tested to secure that all requirements have been dealt with and met. The system is delivered to the customer after the testing has been completed.

5. *Operation and maintenance*, the delivered system is installed at the customer site and also exposed of practical use by the customer. Errors that are not found during the development in earlier phases of the lifecycle are corrected here in the maintenance phase. Improving system units and increasing the services of the system for newly discovered requirements are also covered in this phase. (Sommerville 1982)

The approach practiced by the waterfall model has helped to remove quite a few of the difficulties that were common for software development prior its creation. “The waterfall model has become the basis for most software acquisition standards in government and industry.” (Boehm 1988: pp. 63)

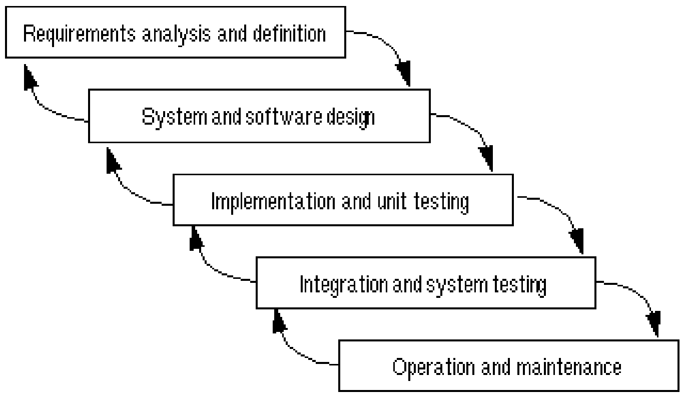
Requirements analysis and definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 3-3. Overview of the waterfall model. (San Diego Mesa College)

3.4 Spiral model

The purpose with the Spiral model was to change the management emphasis to risk analysis since the lack of this in earlier lifecycle models often lead to project failures. This in combination with continuously produced prototypes of the system was meant to reduce risks and problems as early as possible during the development. It is a very flexible model, e.g. it can be accommodated to a variety of other lifecycle models and also provides a hint to which combinations of other models that are most appropriate in a given situation. It is iterative and progresses by the completion of a cycle in the Spiral, which consists of four main blocks. It can be used for iterative- and incremental development and it also supports the use of parallel Spiral cycles, e.g. one for each component in a software product (Boehm 1988).

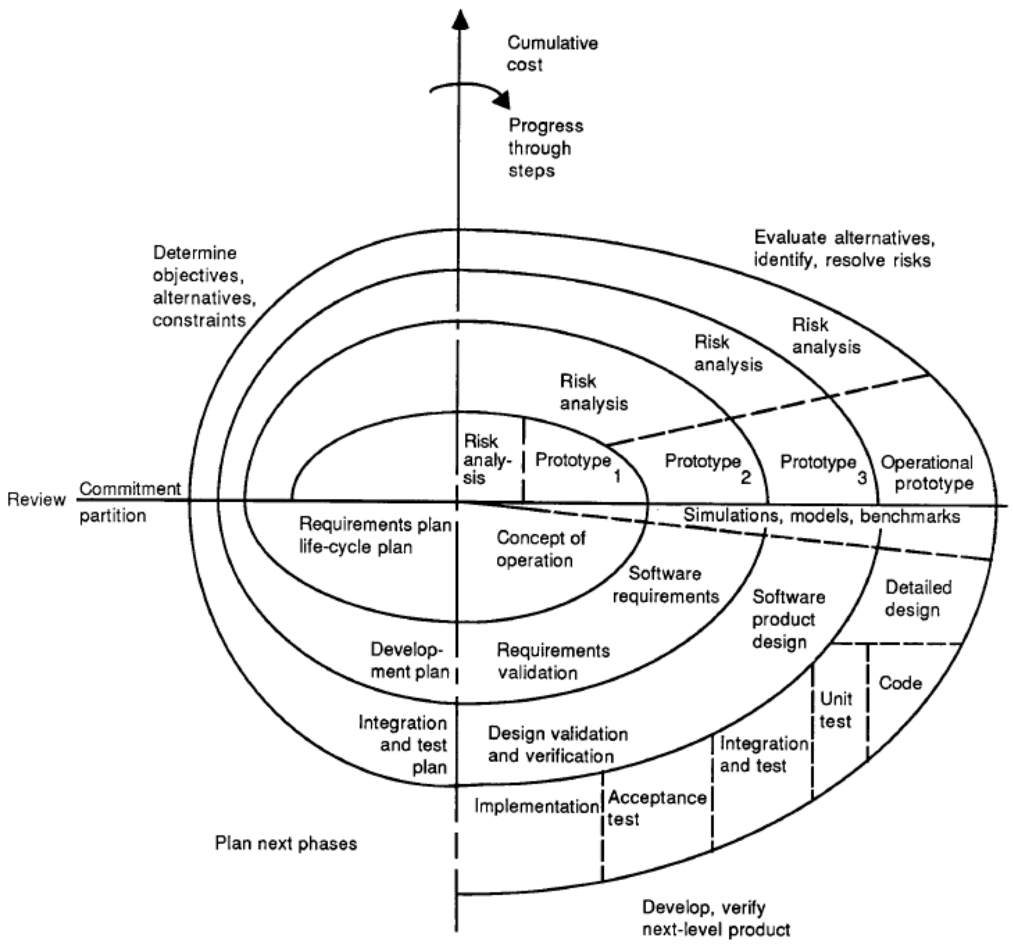

Figure 3-4. Overview of the spiral model. (Boehm 1988)

An overview of the Spiral model is presented in Figure 3-4, the Spiral is initiated in the middle and proceeds clock-wise towards the outside. The four main blocks are as follows:

1. Determine objectives, alternatives, and constraints.

2. Evaluate alternatives, identify and resolve risks.

3. Develop product and validate product and software process definitions.

4. Plan the next cycle.

In the first block the objectives, alternatives, and constraints of the product and software process should be elaborated. The evaluation in the second block refers to the alternatives with respect to the objectives and constraints. Major sources of product and software process risks should also be identified and resolved in this block, and prototypes should also be produced. In the third block the product is developed and both the product and software process definitions are validated. In the fourth and final block the next cycle is planned and a review of the developed artefacts is performed. The management's commitment to proceed as planned should also be secured in this last block (Boehm et al. 1998).

The risk-driven approach that is practiced in the Spiral model allows the model to accommodate any suiting combination of lifecycle models as already mentioned. The selection of a suitable strategy is based on the severity of the risks found and the effectiveness of the techniques to be used to solve them. Risk management also deals with deciding the time and effort that should be devoted to other project activities such as planning, configuration management, and testing. This can vary depending on the number and severity of the risks found (Boehm 1989). This model is more descriptive compared to the Waterfall model since it can be adjusted to the current situation of an ongoing project.

Each loop in the Spiral in Figure 3-4 represents a phase of a software process, e.g. the innermost loop might cover system feasibility, the next system requirements, and the next system design and so on. The phases in the Spiral model are not all required to be performed, it is up to management to decide how to structure their project into phases (Sommerville 1982).

3.5 Summary

In this chapter a definition of the software process lifecycle model was presented along with descriptions of two well known models, namely the waterfall- and spiral model. The main functions for the lifecycle model are to determine the order of stages involved in software development and evolution, and also establish the criteria for how the development should move from one stage to another.

The characteristics of the waterfall model are that software development is carried out through predefined sequential phases and that it is inflexible, a new phase should not be entered before the prior one is completely finished. The spiral model on the other hand is an iterative model that focuses on flexibility and risk management. The phases described in the spiral model are not all required to be performed, it is very adaptable and can be adjusted to fit the current situation of an ongoing project.

4 AGILE SOFTWARE DEVELOPMENT

4.1 Introduction

This chapter is divided into three different parts, the first part starts with the background of Agile software development to get a better understanding of why the Agile manifesto and its principles was developed and a definition of these are also presented. In the second and third parts XP and DSDM are defined and this is done by presenting their lifecycles, practices, and roles, these three parts are included in the description to give an overall presentation of these software processes.

In the summary our identified characteristics of Agile software development with focus on the lifecycle are highlighted, these characteristics are true for both XP and DSDM. The purpose of this is to establish a foundation to compare these against how the interviewed software companies perform their software development.

4.2 Background

In the early years of software development many organizations and developers were developing software without having a plan to follow. The design where made by a number of short term decisions. This worked all right as long as the systems where small and not too complex, but as the systems grew new features were more and more difficult to add (Humphrey 1989).

We lived with this type of development for a while and then came the time for different methods, that is to say software processes. Examples of these software processes are RUP and CMM (Charette 2001). The aim with them was to make the software development more predictable and more efficient, this was made with a strong emphasis on planning. They have been criticised for being bureaucratic, to slow down the development and are often referred to as heavy methods (Fowler 2000).

As a reaction to these heavy methods have a number of light methods turned up during the past few years. Warsta et al. (2002) suggests that this is an answer to the eager business community asking for lighter and faster methods. These methods try to compromise between too much software process and no software process at all.

In February 2001 seventeen experts met and discussed the growing field of what used to be called lightweight methods and this was the start of the Agile Alliance. Some of these methods are XP, DSDM, Crystal, SCRUM, Adaptive Software Development (ASD), and Feature-Driven Development (FDD). The result of this meeting was a manifesto with twelve belonging principles for Agile Software Development (Fowler et al. 2001). The idea of this manifesto was to capture the common ground of these methods and act as a rallying cry to the software industry (Fowler 2002).

The new Agile software processes have engendered an intense debate about Agile software development versus rigorous software development (Highsmith et al. 2001). This debate has just started and there is still much research to do. McCauley (2001) suggests that, since the first evidence of the effectiveness of the Agile software processes seems to be positive and the evidence is growing, we should at least consider them.

The Agile manifesto and its principles are actually only some practices that have been uncovered not invented (Cockburn 2002). The principles are sometimes refereed to as a formalization of practical, ‘common-sense practices’, that quite successfully have been used on many real world software projects (McCauley 2001). Orr (2002: pp. 11) states that the Agile manifesto are “a set of very simple and straightforward common principles”. This is a common view of the manifesto and this makes you wonder if the manifesto is not already used.

4.2.1 Manifesto

The Agile manifesto that the seventeen experts agreed on consists of four central values. They say that there are values in the items on the right but that the items on the left have even more value. The statements are:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan (Beck et al. 2001, Cockburn 2002)

First, the Agile movement emphasizes on the people instead of the software process. Although they do not say that a team can manage without a software process (Cockburn 2002). Interaction between the individuals is also being valued. The interaction matters and better interaction between the people is beneficial for both software process-centric development and chaotic development (Cockburn 2002).

Second, Cockburn (2002) states that working software, that is running, is the only thing that can tell what a team really has built. Documents should only be used as hints to guess what the future will look like. A vital objective in the Agile manifesto is to continuously deliver working software to the customer. The code should be simple, straightforward, as advanced as needed, and just as much documentation as required should be used (Warsta et al. 2002).

Third, in Agile software development the developers and the customers who want the software should be seen as ‘us’, not ‘us’ and ‘them’ (Cockburn 2002). A good relation to the customer can save a contract situation and it can sometimes make a contract unnecessary. The negotiation process should be seen as a way to make the relationship with the customer more solid.

Fourth, a plan is always useful, as long as it is up to date. Changes to a plan must be made as fast as possible, otherwise is it useless. Relatively short development phases are used to be able to react as fast as possible to changes. The development group must be able to make decisions about changes concerning a product (Warsta et al. 2002).

The Agile Alliance does not have all the answers. They have four statements as their manifesto and these statements are uncovering better ways of developing software and helping others do it (Fowler et al. 2001). Boehm (2002) illustrates, in Figure 4-2-1, various methods along a spectrum of increasing emphasis on planning:

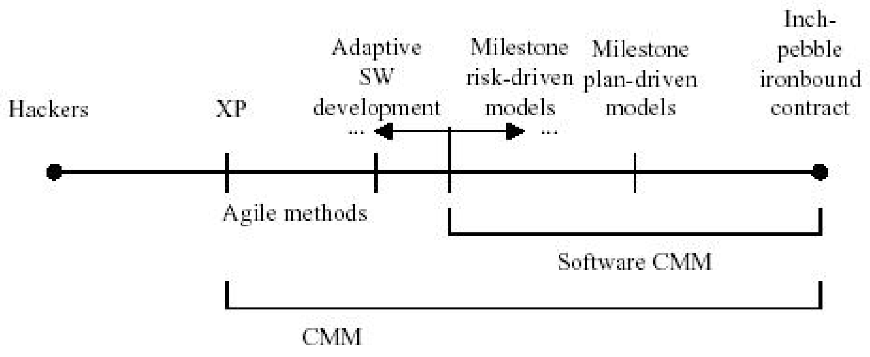
Hackers
XP
Adaptive SW development
Milestone risk-driven models
Milestone plan-driven models
Inch-pebble ironbound contract
Agile methods
Software CMM
CMM

Figure 4-2-1. The planning spectrum.

This spectrum shows that it is almost impossible for only one method to fit all projects. Software projects vary wildly in, size, complexity, risk, criticality, technology, and cultural constraints, and many other key variables. Agile methods will probably fit some software companies but not everyone. It is time to stop the methodology crusades (McCormick 2001). The 'Silver bullet' that everyone are looking for do not exist. "What's needed is not a single software methodology, but a rich toolkit of process patterns and "methodology components" (deliverables, techniques, process flows, and so forth) along with guidelines for how to plug them together to customize a methodology for any given project." (McCormick 2001: pp. 110)

4.2.2 Principles

The next level of the statements in the manifesto is the principles. They are developed to support the manifesto and together with the manifesto give enough information to build your own Agile work habits (Cockburn 2002). The 12 principles are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

3. Working software is the primary measure of progress.

4. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

5. Business people and developers work together daily throughout the project.

6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

8. The best architectures, requirements, and design emerge from self-organizing teams.

9. Continuous attention to technical excellence and good design enhance agility.

10. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

11. Simplicity—the art of maximizing the amount of work not done—is essential.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly. (Cockburn 2002).

4.3 XP

"Software development fails to deliver, and fails to deliver value. This failure has huge economic and human impact. We need to find a new way to develop software" cited from (Beck 1999: pp. 3).

XP was a result of a project at DaimlerChrysler using new concepts in software development. It consists of a set of key principles and practices affecting all parts of development. The term 'extreme' comes from taking these common-sense principles and practices to extreme levels (Beck 1999). This software process improves a project in four essential ways: communication, simplicity, feedback and courage. You need to improve communication. You need to seek simplicity. You need to get feedback on how well you are doing. And you need to always proceed with courage.

There are some limits of when to use XP. The exact limits of XP are not clear yet but examples of limits are big teams, distrustful customers, technology that does not support changes (Beck 1999). The size of a team should be between 2 and 12 but projects of 30 have been reported successful (Wells 1999). One of the reasons for the creation of XP was the domain with frequent changes of requirements. Another requirement is testability. You must be able to create different tests like automated tests and functional tests.

4.3.1 Software process lifecycle

The lifecycle of XP consists of six phases: Exploration, Planning, Iteration to Release, Productionizing, Maintenance, and Death. The lifecycle is illustrated in Figure 4-3-1. This lifecycle has very much in common with the Spiral lifecycle model since both of these are iterative and focus on finding risks as early as possible, they also use prototyping.

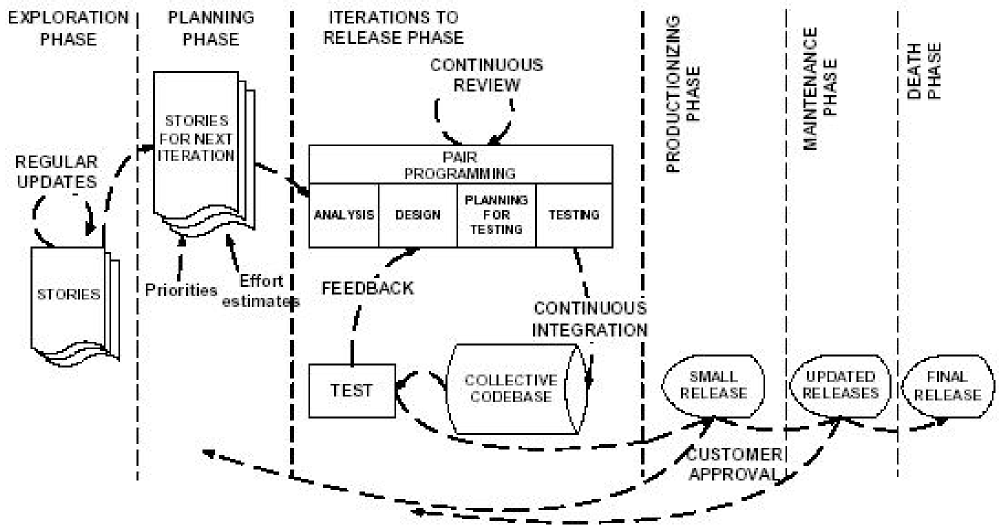

Figure 4-3-1. Lifecycle of the XP process. (Warsta et al. 2002)

The phases of XP are presented in the following according to Beck (1999).

The **Exploration Phase** should be out of the way as quickly as possible. XP says that not to be in production is to be spending money without making money. However, you have to believe that you can go into production before doing it. The phase starts with that the customer writes the user stories and the programmers estimate them. At the same time the programmers explore possibilities for the system architecture and try to build at least one prototype. Experiments with the performance limits are made to eliminate as many performance risks as possible. This phase can be between a few weeks to a few months, depending on how new the technology or domain is for the programmers.

The next phase is the **Planning Phase** and its purpose is to get the customer and programmers to agree on a date of the first small release. A priority order and estimations for the stories will be made with a planning game. The goal of this phase is to get as much software produced as possible. The first release should be delivered after between two to six months. The planning phase should take a few days.

The **Iteration to Release Phase** breaks the schedule set in the planning phase into one- to four-week iterations. A number of test cases are produced for each iteration and these are all carried out at the end of the iterations. The first iteration is chosen to create the architecture for the whole system. The sequence of the rest of the iterations is chosen by the customer. The plan is changed when deviations from the plan is detected.

In the **Productionizing Phase** the feedback cycle is tighten up from e.g. three-week iterations to one-week iterations. This increases the need for testing and parallel testing is often used in this phase. The performance of the system may also be tuned up. New changes are still allowed before the release but some of these should be put in a list for the further releases.

The **Maintenance Phase** is the normal state of an XP project. The released system is maintained to be running, simultaneously is new functionality produced. Team members are replaced or removed and new members are added to the team. Every new release starts with an exploration phase. Much larger refactorings will be possible since new releases are not as dangerous as the first release was. New releases may add new technologies or even change the architecture.

The last phase in an XP project is the **Death Phase**. This phase occur when the customer can not come up with any new stories. The death of a project may also occur if the money has run out or the team just can not deliver what is needed. The system is documented for facilitate future changes and a farewell party is held.

4.3.2 Roles and responsibilities

XP consists of several different roles with different tasks and purposes. These roles are used during the software process and can be changed whenever they are needed. People in an XP project can have more than one role. Beck (1999) describes seven roles in an XP project and these are presented in the following.

Programmer is the role that defines tasks from user stories and estimates them. They constantly try to make the programs larger, simpler and faster. Unit tests are also implemented by the programmers to verify the program.

Customer writes the user stories and sets implementation priority to them. Any questions about the user stories are answered by the customer and to verify the system the customer also writes the functional tests. The customer may or may not be an end-user but they have authority to decide answers to the questions about the user stories.

Tester helps the customer to choose functional tests. They are also responsible for implementing and running the functional tests. The people affected by results that are divergent from the expected outcome must be informed by the testers.

Tracker monitors the working progress of the programmers and takes action if things seem to be going off track. Halfway through an iteration a tracker should be able to tell if anything needs to be changed to follow the current plans or a new plan is needed.

Coach is responsible for the software process as a whole. What practice that might help when problems occur, what the ideas behind XP are, and how to relate these to the project. A coach should help the team to reveal mistakes without too much interference and steering.

Consultant is an external member possessing specific technical knowledge in a specific area. The consultant helps the team solve their problems. Since everybody is changing partners all the time, there is little chance that only one or two people will understand a certain part of a system. This is one of the strengths of XP, but it can also force a team to procure a consultant with deeper technical knowledge in some specific areas occasionally.

Big Boss reacts to differences from the plan. To do this, a big boss communicates with the XP team to determine the current situation. If an XP team does not produce what they should, a big boss can step in and help them.

4.3.3 Practices

"We will rely on the synergies between simple practices, practices that often were abandoned decades ago as impractical or naïve" cited from (Beck 1999: pp. 3).

XP proposes 12 software development practices to increase productivity while maintaining quality (Maurer et al. 2002). These practices support each other and the weakness of one practice will be overcome by the strengths of other practices. The 12 practices will be summarized in the following.

The Planning Game is used to chart the scope of a project and to prioritize. This is made with a dialog between the business people and the technical people. Business interests can be represented by various people, including marketing personnel or a customer's employee. Whoever it is, that person has to have knowledge to decide about the scope of the project, what to be prioritized and the dates for releases. The technical people are responsible for the decisions about estimates, consequences and how the team will work and be organized (Maurer et al. 2002).

Small Releases of the developed software is practiced in XP**.** Each release should be as small as possible and still produce a useful software system that generates business value for the customer (Beck 1999). It is better to plan a month or two between each release than six months or a year. Short cycles are used to reduce the risk that a project fail to produce business value to the customer. Planning problems and the problem with changing requirements are also reduced with small releases.

Each XP project has a **Metaphor** to guide all development by describing how the system should work. It represents a coherent view of the system that makes sense to both the business- and technical-people (Maurer et al. 2002). The metaphor could be seen as an overall architecture that is easy to communicate and elaborate.

XP advocates **Simple Design** since they assumes that the future is uncertain and do not want to create a design for the future. The focus lies on solving today's problems and every piece in the design must be able to justify its existence. The right design in XP can run all the tests, has no redundancies, and has the fewest possible classes and methods (Beck 1999).

Testing is an essential part of XP. Especially the automated tests, a feature without an automated test does not exist. The programmers write the unit tests and the customer writes the functional tests. The functional tests are then used as acceptance tests to verify that the program does what it is supposed to do (Maurer et al. 2002). The actual code in XP is written after the test cases are written.

The process of changing a software system in such a way that it does not alter the external behaviour of the code and yet improves its internal structure is called **Refactoring**. This is made to keep the design as simple as possible at all times (Bailey

2002). The changes of the structure are verified with automated tests which also help the programmers to get feedback on the changes.

All the production of code is written with two people using one computer, **Pair Programming**. One programmer, the driver, has control of the keyboard/mouse and creates the code, the other, called the observer, is continuously assuring quality, i.e. he/she is watching, trying to understand, asking questions, looking for alternative approaches, helping to avoid defects (Nawrocki et al. 2001).

Collective Ownership for the code produced is used in XP. Everybody in a XP project takes responsibility for the code in the whole system. Any improvements or new ideas can be added anywhere in the code (Beck 1999). This can be made partly due to the automated tests in XP. Unknown repercussions will be detected by the automated tests and the programmers can modify the code more freely.

Continuous Integration. Changes to the code are integrated at least once a day. The pair programmers are responsible for integrating their own code and automated tests are run to ensure that the system still is working at 100 %. If the tests fail, the pair can undo their changes and start over (Beck 1999).

40-Hour Weeks for the involved project members are very important. Overtime is a symptom of a serious problem in an XP project. The rule in XP is: you can not work a second week of overtime. Two weeks of overtime will be seen as a problem that needs to be solved (Beck 1999).

A customer needs to be available, an **On-site Customer** in XP, to determine and prioritize the requirements. This is one of the few requirements in XP and it helps to improve the software business value (Beck 1999). The programmers can get input from the customer immediately instead of speculating. Quick changes to the focus of the development can also be made when necessary.

Coding Standards keep the code consistent and easy for the entire team to read and refactor. This must be followed since everybody in the team constantly changes partners, refactor other’s code, and changes parts of the system all the time (Beck 1999).

4.4 DSDM

In 1994 the DSDM Consortium was formed by information systems professionals from both small and large organizations, which operated in a wide variety of industries. These professionals together with consultants and project managers from some of the largest Information Technology (IT) companies in the industry formed this not-for-profit Consortium (Stapleton 1997). The work performed by the Consortium later lead to the first version of DSDM, which was released in February 1995 (Howard 1997).

DSDM is a framework of controls for Rapid Application Development (RAD) with guidelines on how to use these controls (Stapleton 1997). It is based on wide experiences and intended to overcome and ease some of the problems related to traditional software development by focusing on early delivery of basic functionality so that workable systems can be quickly produced, thus reducing the time and costs of projects (Howard 1997). The goal for DSDM is to understand the business needs and deliver a working solution as fast and cheaply as possible. “The DSDM philosophy is to do enough and no more.” (Stapleton 1997: pp. 5)

Two very important elements in DSDM that are used in the development are time-boxes and prototypes. Timeboxes are the mechanism applied for handling flexibility of requirements, and this is done throughout the project. The completion date for the project can be seen as an overall timebox for the work that must be done, DSDM then nest shorter timeboxes of two to six weeks within this overall timebox. Each of these timeboxes pass through three phases before they are complete, these phases are investigation, refinement and consolidation. The investigation is to see if the project is going in the right direction, the refinement builds on the review from the investigation and the consolidation ties up any loose ends (DSDM Consortium). Prototyping is a key technique in DSDM as mentioned, this is due to the fact that it support both the developers and the users.

4.4.1 Software process lifecycle

The lifecycle of DSDM is shown in Figure 4-4-1 and this is also known as "the three pizzas and a cheese" (Stapleton 1997: pp. 3). The lifecycle has five main phases and these are preceded by the Pre-Project phase and ended with the Post-Project phase making a total number of seven DSDM phases (DSDM Consortium). The lifecycle of DSDM has many resembling characteristics with the Spiral lifecycle model. Both lifecycles are iterative and use prototypes, they also focus on identifying risks as early as possible in the lifecycle. The five main phases are: Feasibility Study, Business Study, Functional Model Iteration, System Design and Build Iteration and Implementation.

The first three phase's pre-project, feasibility- and business study are performed in a sequential approach. "The primary aim of the earlier stages of DSDM is not only to assess the feasibility of the project and the suitability of DSDM for that project, but also to achieve consensus as to the short-term aims of the project." (Barrow et al. 2000: pp. 97)

The main purpose of the pre-project phase is to ensures that only the right projects are started and also that they are correctly set up. When it has been decided that a project will be carried out, the planning for the feasibility study is performed and then the project is proper started when the feasibility study begins. The first two main phase's feasibility- and business study are done to set the ground rules for the rest of the development, that is performed iteratively, and therefore these must be completed before any further work is carried out on a given project. How the last three main phases overlap and merge is left to a particular project to decide. The main purpose of the post-project phase is that it should keep the solution developed operating effectively. Due to the iteratively fashion of DSDM, maintenance can be seen as a continuance of the development. (DSDM Consortium)

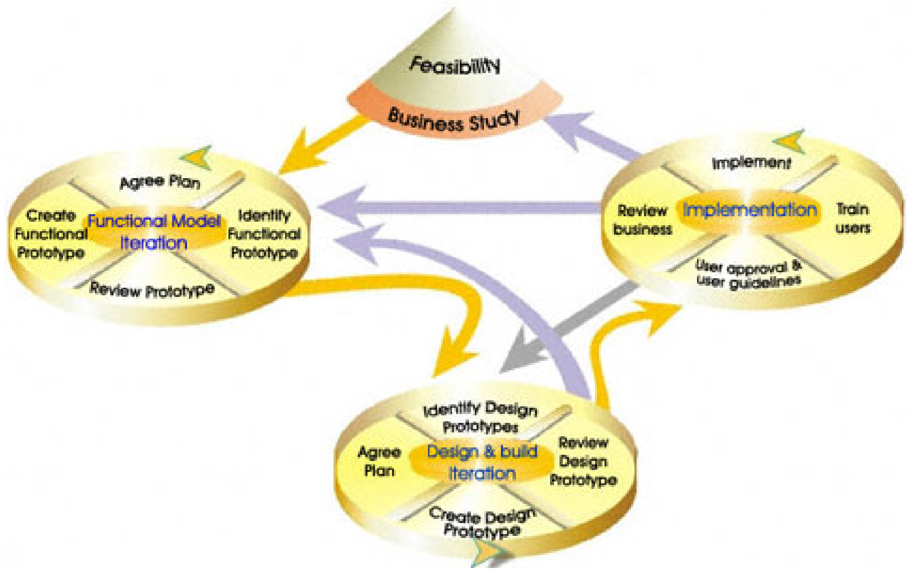

Figure 4-4-1. DSDM process diagram. (DSDM Consortium)

The five main phases of DSDM are presented as follows by Stapleton (1997).

The **Feasibility Study** in DSDM is not as large and comprehensive as a traditional feasibility study, but the normal considerations are still present. This includes for example a definition of the problem domain and also answers to a number of questions, such as ‘Is it technically possible to produce the system?’, ‘How are the current business processes affected, is this acceptable?’, and also of course ‘Is the system worth doing?’. Another very important question is, ‘Is DSDM the best possible software process to use for building this system?’. This question is important because the way DSDM carry out its development might not fit all organizations or it might be very difficult to produce some specific products in this manner. According to the DSDM Consortium a proper project for DSDM is a project were all the classes of users who will use the end result are possible to identify so that their views and ideas are collected and used throughout the entire project. Large projects should be able to be broken down into smaller components for development by parallel teams or for incremental deliveries.

The feasibility study should last no more than a few weeks and generate three different products, these are a feasibility report, outline plan for development, and a fast prototype. The prototype thus is an optional product. The feasibility report will include all ‘usual’ topics but not in detail, it should present enough information so that decisions about risks etc. can be based upon it. The outline plan will include information that supports that the desired outcome is possible to achieve.

The **Business Study** works as a foundation for the remaining phases and the work performed in them, this phase is as the feasibility study just performed during a relatively short period of time. It should give enough understanding of the business and technical constraints to let the project carry on safely. The most important activity in this phase is as the name suggests, obtain an acceptable understanding of the business process and also their information needs. This is done by close collaborations with the customer where workshops play an important role.

The series of workshops performed will lead to the creation of the business area definition document in which the business process, related information, and also the different classes of users who will be affected by the introduction of the system will be identified. Another important document produced during this phase is the system architecture definition in which the development, target platforms, and the architecture of the system are described. The outline plan produced during the feasibility study is refined to produce the outline prototyping plan. In this plan all prototyping in the next two phases are covered, this includes both prototyping strategy and the configuration management plan.

In the **Functional Model Iteration** phase the focus lies on refining the business aspects of the system, this means building on the high-level functional and information requirements found in the previous phase. In this and in the design and build iteration phase four activities are performed repeatedly to move forward with the development. In each cycle the following activities are performed:

1. Identify what should be done in the cycle.
2. Agree upon how to achieve it.
3. Perform the task.
4. Check to see if it was properly done by reviews, tests or demonstrating a prototype.

During this phase both standard analysis models and software are produced, these contain the major functionality and should also satisfy some of the non-functional requirements. These two are built side by side, but with more focus on the analysis models in the beginning. The findings in one cycle will then be used in the next to evolve both the analysis models and prototypes that possibly can be delivered further on. Another important activity in this phase is testing, the focus lies on testing what the components do and if they will form usable sets of functionality.

Other activities during this phase are prioritised functions which include producing the details of the high-level functions and prioritize them, risk analysis of future development, record all non-functional requirements, and produce a functional prototyping review document which includes comments from the users regarding the prototypes.

The **Design and Build Iteration** is the phase where the system is further developed so that it can safely be handed over to the users holding the quality standard that they requested. The testing in this phase focus on the non-functional requirements, but also unit-, integration- and system testing are performed. The way the work is conducted is according to the description in the previous phase, thus all development is carried out by performing the four activities mentioned repeatedly.

The main product produced in this phase is the tested system. The reason for that testing is not presented as a distinct activity in the DSDM lifecycle is that it should be performed throughout both the functional model iteration and the design and build iteration phase.

In the **Implementation** phase the system is moved from the development environment to the operational environment, this includes both handing over the system and training the users. This can be done iteratively if the system is delivered to a user that is located in a number of different locations over a certain period of time, otherwise the phase is just performed in a single iteration. Two very important documents are produced during this last phase, the user manual and the project review document in which the achievements of the project is summarized in terms of the short-term goals. It reviews in particular the requirements that have been identified and assess the system in relation to those requirements.

When this phase is completed there are four possible outcomes for the project to proceed:

1. There are no more requirements to satisfy so the work is done.
2. Return to the business study for dealing with a business functionality that was not attended previously due to time shortage.
3. Return to the functional model iteration for attending lower priority functionality that was left out because of the time scale.
4. Return to the design and building iteration for attending an area of concern that was omitted due to time pressure.

4.4.2 Roles and responsibilities

DSDM defines 15 different roles, these are for both developers and intended users of the system. A project team in DSDM is kept small, it usually consists of two to six people. Two is a minimum to ensure that one IT person can perform the technical work and one user to satisfy the business needs. Six have been agreed as a maximum because the RAD process has difficulties to be carried out if this limit is passed. One project can have more than one team and one person can hold more than one role (Stapleton 1997). One very important ingredient to a successful DSDM project is people, how these interact and communicate is crucial. Stapleton expresses this as follows, “DSDM is about people, not tools.” (Stapleton 1997: pp. xiv)

The following roles are the ones presented by Stapleton (1997) as the most significant ones.

Technical Coordinator defines the system architecture, and is responsible for ensuring the technically consistence in the project and the quality. This person is also responsible for checking that technical controls such as configure management work as planned.

Senior developers and **developers**. DSDM do not make any distinction between different IT roles, such as analysts, designers, programmers, etc. All IT personnel are categories as senior developers or just developers, the distinction between these roles are based on experience of the work to be performed and the experience of working in a RAD environment

Visionary is a user role that acts as the project champion. This person is largely responsible for the initial concept, and also works as an overall advisor for the rest of the project.

Ambassador User represents the business area in the DSDM team, provides key inputs to business requirements and design sessions. This person can also answer developer questions quickly and authoritatively during the project.

Advisor User can be any one who has an interest in the final system, this role is involved on ad-hoc basis. This role was created since the Ambassador can not cover all user views.

Executive Sponsor is a person that has the financial responsibility and is the ultimate decision maker.

4.4.3 Practices

DSDM is based on nine practices that work as a foundation for the development, and these are applied as appropriate in the different parts of the software process. The practices which are called principles in DSDM are presented as follows by the DSDM Consortium.

Active User Involvement is imperative. The users should be active participants during the development. If this is not applied, delays will probably occur and the users could feel left out.

The **Team Must be Empowered** to make decisions. Teams in DSDM consist of both users and developers, and they must be able to make decisions themselves as requirements change. They should not be required to discuss all details with higher-level management. This could concern certain levels of functionality, usability etc.

The focus is on **Frequent Delivery of Products**. The DSDM team strive to deliver products frequently, the time between deliveries should be kept short, from one until a couple of weeks. This is to make it easier to decide the most important activities for producing the right product.

Fitness for Business Purpose is the essential criterion for acceptance of deliverables. The focus for the DSDM team is to deliver the most important business requirements within the required timeframe.

Iterative and Incremental Development is Necessary to converge on an accurate business solution. The system being developed is allowed to grow incrementally, so that the DSDM team can take user views from one iteration and feed it into the next one to steer the solution to better fit the users' needs.

All **Changes** during development are **Reversible**. The development must be in a known and understandable state all of the time, this will ensure that the DSDM team has full control of the development of the product.

Requirements are baselined at a **High Level**. The purpose and scope of the system should be agreed upon by the involved parties, this should be on a level that allows for a more detailed investigation later of what the requirements will entail.

Testing is Integrated throughout the lifecycle. In DSDM testing is not treated as a separate activity since it should be performed along with reviews by both developers and users throughout the software process lifecycle.

Collaboration and Cooperation between all stakeholders **is Essential**. In the beginning of a project the short-term direction must be possible to decide quickly without any changes needed in the planned procedures. The relation between all involved stakeholders is extremely important and this is not just between the developers and the users, but also other involved parties such as resource management etc.

4.5 Summary

The lifecycle characteristics of XP and DSDM which we have identified are presented below and also how these are implemented by these two Agile software processes. These characteristics will later be used as foundation when analysing and comparing how the interviewed software companies perform their software development in relation to these Agile software processes. Since the roles defined by the Agile software processes does not affect the software process lifecycle greatly, we have decided to not include these in the analysis.

- Iterative development
- Active customer collaboration
- Frequent releases
- Continuous testing

The **iterative development** in XP is started in the iterations to release phase where the analysis, design, implementation, and testing are carried out. During the productionizing phase are the iterations tighten up from e.g. three-week iterations to one-week iterations. The first iteration should create the architecture for the system, the sequence of the remaining iterations is chosen by the customer. In XP the development team then returns to earlier phases from the productionizing and maintenance phases to deal with further user stories and respond to feedback from the customer. In DSDM the iterative phases are functional model iteration, design and build iteration, and occasionally also the implementation phase. The activities performed iteratively in these phases are analysis, design, implementation, and testing. The system being developed is allowed to grow incrementally, by doing this user feedback from one iteration can be used in the next one to steer the solution to better fit the needs of the users. It is also possible to return to earlier phases from the three iterative ones for responding to changes or further feedback from the users.

Active customer collaboration in XP is supported by an on-site customer that always is available and also is responsible for determine and prioritize the requirements. DSDM requires active user involvement for avoiding delays and misunderstandings, so communication with the user should be very frequent and open throughout the entire project. This characteristic affects the lifecycle since this communication can be used to take immediate decisions that could lead to changing the plan or the requirements etc, instead of this procedure taking a number of days that might lead to that the project is falling behind schedule.

To reduce the risks of failing with producing business value to the customer, planning problems, and problems with changing requirements XP uses **frequent releases** or small releases as they call them. Each of these releases should be as small as possible but still produce a system that generates business value to the customer. DSDM strive to deliver products frequently, the time between deliveries is always kept short, from one until a couple of weeks. This is to ease the decision on deciding the most important activities for producing the right product.

Continuous testing is essential in XP, the actual source code is written after the test cases are done. XP advocates the importance of automated tests which are used as much as possible. Unit tests are written by the programmers and the customer writes the functional ones. DSDM does not treat testing as a separate activity since it is performed with reviews by both users and developers throughout the entire lifecycle. The mechanism used for handling all these tests are timeboxes, in which testing is performed continuously throughout the entire software process.

5 INVESTIGATION APPROACH

5.1 Introduction

The purpose of this master thesis is to find out how much of the lifecycles used by non-Agile software companies that are similar compared to the ones used in the Agile software processes XP and DSDM. Since Agile and many of the Agile software processes are quite new and only few research reports about these have been made, we thought that a 'real life' example would be the best way to investigate this. Hence, this chapter will try to answer which investigation approach that is best suited for this master thesis and also describe the approach that we chose. Our experiences of this method will also be expressed in this chapter.

5.2 Quantitative and qualitative methods

Qualitative methods focus on naturally occurring, ordinary events in natural settings, and on what 'real life' is like. They are also suited for locating the meanings people place on the events, processes, and structures of their lives: their "perceptions, assumptions, prejudgments, presuppositions" (Miles 1994). Quantitative methods try to answer the question how much, be more precise than qualitative methods, and generalize the collected data (Svenning 1996).

An example: The quantitative method would try to find out how many are using a certain software program but a qualitative method would try to find out e.g. what kind of programs that are used, how they are used, how they work, and if they will be used in the continuation.

Both qualitative and quantitative methods have their advantages and drawbacks. Quantitative methods often sort the data into categories that might not 'fit' in order to make meaning. The selection, when making a quantitative research, must be statistical representative since the frequency has to be mentioned (Trost 1993). Qualitative methods, on the other hand, sometimes focuses too closely on individual results and fails to make connections to larger situations or possible causes of the results. An advantage with qualitative methods is that they have strong potential for revealing complexity, since the data is rich and holism (Miles 1994). Qualitative methods also produce a wealth of detailed information about much smaller number of people and cases which increase the understanding of the cases and situations studied but reduce the ability to generalize it. By contrast, quantitative methods make it possible to measure the reaction of a great many people to limited set of questions. This gives a broad set of findings that can be generalized (Patton 1980).

Many researches agree that qualitative and quantitative research methods need each other more often than not (Miles 1994). However, because typically qualitative data involves words and quantitative data involves numbers, there are some researchers who feel that one is better or more scientific than the other. The use of a quantitative method can sometimes be a requirement when writing a report for the science community, general public, or a public institution (Kvale 1997). It is also sometimes used to make a report appear more trustworthy than a qualitative report with descriptions and interpretations. Qualitative methods on the other hand, have often been advocated as the best strategy for exploring a new area, developing hypotheses, or finding new discoveries. Another difference between them is that quantitative methods are deductive and qualitative methods are inductive. In qualitative methods, a hypothesis is not needed to begin researching. However, all quantitative methods require a hypothesis before research can begin (Miles 1994).

This is an ongoing debate and some researchers even think that they can be used in combination only by alternating between methods: qualitative research is appropriate to answer certain kinds of questions in certain conditions and quantitative is right for others. And some researchers think that both qualitative and quantitative methods can be used simultaneously to answer a research question (Holme 1986).

5.3 Method used in this master thesis

The method that we have chosen to use in this master thesis is the qualitative method. It helps a researcher to focus on what the 'real life' is like and also to learn more about this in a scientific manner. Thus the qualitative method is the method that best maps onto the situation of this thesis. The starting point of this master thesis is that much of the Agile manifesto and its principles are already used in the industry. Hence, even non-Agile software companies must follow a majority of these guidelines. This has been investigated by conducting five interviews with software companies that do not use any Agile software process.

Five interviews with five different software companies was made after most part of the literature study was finished and these interviews was accomplished with the help of an interview guideline, which can be found in chapter 10.2 "Interview guideline used in the survey", with approximately 10 categories and a number of sub categories. Some of these categories were the person being interviewed, a recently performed project, communication, the developers, documentation, and deliveries. Most of these emphasis on the lifecycles of XP, and DSDM, and also the Agile manifesto and its principles to facilitate the analysis. All questions were open-ended and each interview took between 45 and 60 minutes per person. Every interview was made face-to-face due to that a face-to-face interview receives answers of much higher quality than interviews or questionnaires performed by phone or email (Patel 1987).

All interviews were then transcribed as the first step of the analysis. The next step was a sentence concentration and a categorisation of the transcribed interviews. Sentence concentration imply that the sentences expressed by the interviewed persons will be more concise formulated and categorisation imply that the interview is divided in different categories (Kvale 1997). From this a summary of each interview was written which later was sent to each project manager for feedback. All this was made to get as much as possible out from the interviews and if possible also some new interpretations or new discoveries.

5.4 Experiences from the qualitative method

The quality of the interviews has varied in this master thesis, some of the interviews have been easier to interpret than others but the result of them resemble each other. The reason for the variety in the quality of the interviews may depend on many different factors e.g. that we are inexperienced interviewers and that some of the interviewees seemed to be nervous of being recorded on tape. Despite this we are glad that we used face-to-face interviews, as mentioned in section 5.3, since this made it possible for us to adapt the interviews to the companies' conditions. An investigation with e.g. questionnaires instead of interviews could have lead to misinterpretations of the collected data since it is very difficult to understand exactly what a person mean when questions are answered just in writing. The qualitative method has reduced the number of conclusions that we have made because the collected information can be interpreted in so many different ways but it has also made our conclusions more trustworthy since the qualitative method focus on getting a ‘real life’ picture of the reality. The conclusions we have drawn after analysing the interview material has been made as objective as possible, but these are of course influenced by our view points and suppositions.

5.5 Summary

The methods presented in this chapter are the qualitative and quantitative method, this was done to find out which method that was the best one to use for this master thesis. Both methods have advantages and disadvantages but only the qualitative method focus on what ‘real life’ is like and this maps best onto the situation of this thesis. The qualitative method used was performed by conducting five face-to-face interviews with open-ended questions. All interviews were then transcribed, sentence concentrated and categorised. All this was made to get as much as possible from the interviews and also give us some new information about how non-Agile software companies conduct software development.

We have got a lot of experiences by using this method and some of them are that face-to-face interviews make it possible to adapt the interviews to the companies’ conditions, the information collected with this method can be interpreted in many different ways, and that it is very difficult to be completely objective since our view points and suppositions influence the final result.

6 THE SURVEY

6.1 Introduction

In this chapter we will present the reason for choosing the software companies that we did for our interviews, the software companies, and also the results from the interviews. To make the interviews more concrete and be able to find out how these software companies really work instead of how they should work according to possible documented software processes, we focused on a specific project that they had performed recently. Overall descriptions of these projects are also presented to make it easier to understand the actual interviews.

6.2 Selection

The selection of the software companies, projects, and interview persons at the software companies did not work out exactly as we planned it from the start. Our first idea was that we wanted to interview software companies that were located in the south parts of Sweden, with roughly the same size, and produced similar software systems for the same category of users. After a while we realized that it would be quite difficult to find such group of software companies, this was because we had problems finding software companies fitting our criteria and also that a number of the software companies we contacted had their software development divisions located abroad or northward in Stockholm or Gothenburg. So we then decided to completely reconsider our criteria for the selection and instead focus on software companies which developed different kinds of software systems and which also had varied number of employees, by this change we got a wider spread investigation than we first planned since we included different kinds of software companies instead of similar ones. When this was done we were able to find all the software companies that we needed and these were also very positive to participate in our investigation.

When selecting the projects to discuss during our interviews we did not specify any strict requirements that these projects had to fulfil, this was not done since we and our supervisors did not believe that it would have an impact on the final result. This was later shown to be a mistake since it was one of the reasons why the analysis of these interviews was quite difficult to perform. The only things we did require was that the standard software project activities analysis, design, development, and testing was performed during the lifecycles of these projects. Some of the software companies asked us if we wanted to discuss projects that covered a product lifecycle, but since all these were still ongoing projects that started a number of years ago we felt that these projects were not exactly what we wanted for our investigation. This was because we could not investigate the entire lifecycle since the projects still were running and we also felt that these were too large, we wanted less comprehensive projects that lasted just a couple of months. This was the reason to why we chose the projects that we finally did for our investigation, these projects also fulfilled the requirements which XP and DSDM have on projects to be able to perform them successfully. At the time for the selection neither we nor our supervisors thought that these companies might use different software processes for different kinds of projects, we realized this first when the interviews were performed. The persons interviewed at these software companies were all project managers for the specific projects, our intention was also to interview persons with great knowledge about how their software companies perform software development during a project e.g. a project manager. Three of the interviewed project managers were women and two of them were men.

6.3 The interviews

6.3.1 Introduction

In this part we present the results from the interviews that we have performed with the five software companies. The letter ‘X’ in these texts represents the interviewed person at each of the software companies.

6.3.2 Company 1

The company was founded in the beginning of the 80’s and has today roughly 80 employees which works with software development, marketing, sales etc. They are active in a vast number of different business areas, but their main focus lies on developing applications for banking, real estate, and insurance.

The project covered in this interview lasted four months, from the fall of 2002 until spring 2003. It was a project which updated one of their systems, this system consists of three different applications one for each area covered. These applications are built on the same platform, components are then used to tailor these for each of the areas. Thus, the focus of this interview lies on the application developed for the real estate market. Updates like this one are performed three times a year. Their customers or at least 95% of them have a service agreement with Company 1, this states that they are obligated to perform three larger updates per year, bug fixes, possibly some smaller updates etc. For this the customers pay a certain amount monthly and get a system that is updated regularly. These delivery dates are agreed upon in advance so all involved personnel at Company 1 and their customers are well aware of when these updates will occur. The reason for choosing three large updates per year is that they have agreed upon this with their customers. This decision is based on the fact that laws, forms, calculations etc. are changed or new ones turn up, so they need to update the system accordingly.

Person X is trained account economist and after the studies X worked for the tax authorities in a revision unit for 4,5 years. X was then employed by Company 1 in 1987 as what they call a ‘product responsible’, this was for a certain product which X today is not involved in at all. X current assignments are product responsible for a real estate product, coordinator and also partial development responsible. X also plays the customers role within the company, this is due to that X handles all customer relations concerning the product X is responsible for. If someone within the project wonders how a specific task should be carried out, X is the person that they consult. X might then contact the costumers if something is unclear and then report back to the project member. Since an update like this one concern a number of different customers, it is possible that all of them can not get all their requirements satisfied for each update. It is up to X to decide what should be included in each of these updates together with developers and customers.

The concerned **customers** in this project differed quite a bit in size, there were customers that used the system on a couple of servers and then distributed it to a large number of offices, and there were also one-man firms that used one version on a single computer. The number of people at Company 1 involved in the project during these four months was about 25 to 30. The programming languages used for the system and then of course in this project were Visual Basic (VB), C and a little C++. The roles used were product responsible that also function as a project leader, developers, testers, and customers.

Company 1 have two different software processes that they have defined and follows, one for updates of standard software and one for projects where they develop completely new products. The software **process** used for update projects consists of a sequential lifecycle with three distinct phases. These phases are requirements-, development-, and the testing phase, during their requirements phase all analysis and design are performed. They always try to follow this software process, some minor changes can be carried out if they believe that they will benefit from doing so. For this particular project the update process was used.

The **requirements phase** is the first one performed by Company 1. The requirements decided for this specific update came from the product responsible, customers, and also developers. Some of these requirements were not included in earlier updates and some were completely new ones, it was thus the product responsible that had the final word of what should be included and what should not. The same procedure was applied for requirements received during later phases. Customer specific requirements could be included in the update for all customers, if the project members believe it to be a good and useful idea for all customers. Otherwise customers can pay extra to get a specific feature developed for their system. All included requirements in the update were time estimated and prioritized, this was done by the product responsible together with the developers. All developers time estimated their own parts that they then developed, these estimations were based on experiences from earlier projects and similar situations. They also decided a date when there should be no more requirements added for the current release, but this was not strictly followed. Requirements that were seen as very important were added even after this date to improve the system. They followed a standard for how the system should look like and be built that they have developed themselves, and this worked as a base for the design. This standard states the position and width of text fields, the interfaces of components etc. It is very important that this standard is followed to keep the appearance of the system intact and to ensure that one component that should be used in different parts of the system looks and works the same in all of these parts.

When the requirements phase was completed the **development phase** started. It continued until four weeks before the delivery took place, during these last four weeks no more development was allowed. As already mentioned new requirements were added during the development phase or old ones reprioritized and even removed to avoid falling behind the time table. The developers also performed different kinds of basic tests of their parts during this phase.

The last four weeks were then devoted to testing and correcting bugs in the **test phase**. The tests performed included both user tests and more advanced technical ones. During these weeks the developers continued to test their parts and components while the testers focused on integration- and system testing. Company 1 wrote the acceptance test themselves which were later used when the system was delivered. When the testing phase was completed the system was delivered to the customers. The customers performed their own user test on the system, some even performed more technical ones to assure that the system corresponded to the agreed requirements. According to the service agreement Company 1 was obligated to correct all faults found by their customers without charge, this included faults in the current update and also older ones. In addition to this final delivery Company 1 usually also distribute some smaller updates via internet during these larger update projects. Feedback from these can sometimes be used for improving the larger update, if it is seen as an important thing for the system.

The **communication** between the project members during the project was very frequent, especially during the requirements- and development phases. They discussed a lot since they all were located in adjacent rooms, also mail, telephones and MSN was used frequently. Another way of communicating practiced by the project group was that they used a specific folder in Microsoft Outlook to which all had access. In this folder everything concerning the current project was kept, there were the requirements specification, error lists, time plans, comments on customer requirements etc. Meetings between the product responsible and selected developers were held once a week during the requirements phase and once every two weeks during the development phase. Company 1 had frequent dialogs with the customer, especially during the requirements phase. Different opinions between the customers and Company 1 were solved by discussions, but in the end it was always money and time that decided what should be included in the update. During the development phase the customers were contacted when changes had to be made to the requirements specification, the communication in this phase was not as intensive as in the previous one. When it was time for delivering the system, personnel at support, sales and also other companies that might be interested in their product were informed that the updated system is now out on the market. Company 1 always sends out a newsletter to all their customers which describe all the features of the new update. The communication with the customers was handled via telephone, mail and meetings. They were always up to date with the progress in the project. Something that Company 1 believes is very important is honesty towards their customers. Their philosophy is that, if you are honest and always describes the current situation truthfully you will gain from this in the long run.

The **project members** had great freedom and were able to affect the outcome of the project. The freedom also implied responsibility towards the rest of the project members and the company. If someone came up with a superior technical solution there was nothing that stopped this solution from being used as long as it was in line with the company standards. There was always someone responsible for the code being produced, but all project members had the right to go in and change in other's code if necessary. Usually there is one person responsible for a certain component, but if it is a large one the responsibility can be shared between several developers. Company 1 always tries to have at least two or three persons that have knowledge about each component in the system to eliminate problems if some one should leave the company. So if possible they let less experienced personnel take some specific parts to widen their knowledge. Thus this is not always possible due to time shortage or that some person may be needed elsewhere. To motivate their employees Company 1 always include all personnel at parties, kick-offs etc. Another kind of motivation is the widening of knowledge as already mentioned, by doing this the employees feel that they evolve and as a bonus from this the company gets better trained personnel. The experience held by the project members were quite extensive, many of them had been working for the company or in similar businesses for more than ten years. During this project the project members put in some overtime during a couple of weeks to make it possible to deliver the system in time, however X did not recall how much overtime it was in total.

To measure the **progress** of the project, they compared the time plan with the requirements specification to know if all planned requirements and activities were performed according to the plan. If they were behind schedule they reprioritized their requirements or even removed some of them after informing the customers. The CASE-tools used during the project were Microsoft Project for planning and SourceSafe for handling different versions of source code and they also saved their documents there. No tools were used for performing tests or handling requirements, this was done by manually. For the design of the user interface VB was used in a straightforward fashion, all items e.g. buttons, text fields were added and then the appearance was evaluated. Company 1 also uses their support division for a number of different things, this division is an enormous resource for the company. They feed the current project with improvement and change ideas throughout its lifecycle, perform some tests, and they also help to write some manuals and help texts for the system.

Changes to the requirements specification were documented, but this is something that they can be better at according to X. All changes should be written in to a supplement document to the requirements specification, and this was then accessible through their shared folder in Microsoft Outlook. This procedure was thus not always carried out, so it is something that they try to improve. To track changing requirements in the project they used these supplement documents, but they did not have any documented tracking between the requirements and the actual source code. As already mentioned, they used SourceSafe for handling different versions of their source code. By doing this they were able to reverse the system to a previous state when something went wrong. Another advantage was that they could supply their customers that were using an old version with the intermediate updates to ensure that the new versions will run correctly. They only used the version numbers for the source code, the documents were saved as their document name and a date. In the documents thus the belonging update and product was written.

According to X they are not very good at planning enough time for writing all **documentation**. Usually almost all documentation is written at the very end of the project and so was also the case for this project. The person responsible for a component was also responsible for producing the documentation for that component. The product responsible produced all documents related to project management and also manuals, some of these were developed together with a developer. They produced documentation for a week and a half, the majority of this was done in the source code. All documents were written in Microsoft Word and most of these were accessible through their shared folder in Microsoft Outlook. The documents that Company 1 produced during this project were a project plan, requirements specification, test documents, user manuals, and meeting minutes for important meetings.

6.3.3 Company 2

The company was founded in the beginning of the 90's and has today roughly 100 employees. Their main focus lies on developing financial systems in Windows environments. Their **customers** that the systems are developed for are small, medium sized and large companies.

The project covered in this interview lasted roughly three years, from the spring of 1999 until fall 2002. The final delivery of the project was planned in the spring of 2002 but was postponed for six months to the fall of 2002. It was a project which updated their financial system for large companies, which is a standard product to a section of customers. Large updates like this one is made approximately every second year and smaller version with new additional functionality and updates to the previous version, the current running one, are released in between. It is built with modules which makes it very adaptable. Specific adaptations to fit their customer are made by the consultant department in the company. The consultant department acted as a customer in this project since they have the contact with the customers. The reason for the update was to raise the technology and make the system component based. They do not have any predefined software process written down for how to carry out a project, but they do follow one as described further on in the interview.

Person X is trained software engineer and after the studies X worked as a software developer for one and an half years. X was employed at Company 2 in 1999 as they call a 'project administrator' and has worked at that position until this project ended. X current position is 'project coordinator'. A 'project administrator' plans the time, the resources and makes for example all documents look the same.

The lifecycle of the software **process** used in the project was sequential, with three phases: requirements-, development-, and test phase, which they have discovered was a disadvantage because many of the developers got surprised how things actually worked when they tested the system in the test phase. This was one of the reasons for the delay of project.

The project consisted of nine **project members** which where reduced to seven during the project. The programming languages used for the system was C++ and Visual Basic for Applications (VBA). The database used was Microsoft Structured Query Language (SQL). Three of the project members were C++ programmers and two were VBA programmers. One project member was responsible for everything concerning reports, for example designing them and connecting them to the database, these things were done since the reports are very important for their financial system. One project member was product manager, which involve the responsibility of making sure that the content and the functionality in a project are correct. Company 2 also has a number of partners that help them develop modules to their system and the ninth project member was responsible for integrating these modules with their modules. It is a role that helps all projects with their integration and the role is therefore not project specific. This project member also wrote some of the technical documentation in this project. One of the two project members that left the project was a VBA developer and the other was the product manager. All tasks that the VBA developer who left was responsible for were handed over to the other VBA developer and the product manger was replaced by a new one. This lead to a lot of over time for the VBA developer and that the freedom of the developers increased a lot during the time when the project was without a product manager.

All new requirements came from the consultant department since they acted as the customer in the project but the basic requirements in the project where taken from the previous version of the system. The consultant department had all the contact with the real customers and all new requirements from the customers came through them. It came new requirements from the consultant department continuously throughout the requirements phase and development phase and all of them were reviewed by the product manager who also decided if they should be added or not. The added requirements were prioritized by the product manager. Only the new requirements were analyzed and designed which made the **requirements phase**, the first phase, very short. Only three months. Object- and sequence diagrams were made of some of the new requirements. The tool used to create these diagrams was Rational Rose. They did not have an all-embracing architecture when creating these diagrams. The time schedule was changed when new requirements dropped in from the consultant department and this was one of the reasons for the delay of the project. All time planning was made by one project member with 15 years experience of the system.

This project member has been involved in developing all modules in the system. They had milestones planned after each finished module, this was to be able to check against the time schedule that they were according to their plan.

When the requirements phase was finished the **development phase** started. All functionality was developed during this phase and the developers made component tests and integration tests. They did not use any coding standards, but they did have both a graphical interface standard and a standard for how to document a component. This phase was approximately two years and the following phase was the testing phase.

The **testing phase** was divided into two parts. In the first part the functionality in the system delivered was still open for changes. Free tests and tests of the new requirements were made. This phase lasted six months which was longer than planned and it made the project even more delayed. The reason for the delay was that the consultant department should have made the tests but they hade a lot of customer assignments at that time so they had to postpone the tests. The latter part of the test phase was divided into three small sections. In the first section the main functions of the system were tested. In the second section was the system tested on a detailed level and the third section tested the graphical interface of the system. All these tests were made by the consultant department and the support department, where two persons acted as testers. The system was delivered with one final delivery which was delivered approximately six months later than planed. The acceptance of the system was made through an acceptance test. This test was carried out by the consultant department and in this all requirements in the requirements specification were included.

The **communication** in the project was handled via meetings, phone, and person to person discussions. All project members participated in a project meeting every morning. These meetings lasted no longer than half an hour and during them all current problems and discussions, if they had any, were brought up. Most of the problems and discussions were handled during the morning meeting but they could also discuss freely during the course of the day since they were seated very close to each other. A project meeting was also held once every week. This meeting brought up the status and the **progress** of the project, which was measured by how many percent per activity that was finished in relation to the time schedule. It was X who followed up the progress in the project. The planning of the project was made in Microsoft Project.

The system consisted of approximately 5 modules and roughly two meetings per module were held with the consultant department during the requirements phase. These meetings were held to see if the requirements were understood correctly and to explain those that were not. The contact with the consultant department almost stopped during the development phase and was not brought up again until the test phase started. They got help from an external consultant with one part of the system that was reconstructed quite a lot. The reason for this help was that they knew that the consultant was very experienced in this area and that no one in their company knew much about it.

The members of the project had great freedom and were able to affect the outcome of the project. This freedom resulted in that the developers sometimes changed things without consulting these changes with someone else which lead to that some things had to be remade. Each developer was responsible for at least one module and no other project member was allowed to go in to their code and change anything. This makes them very vulnerable if someone wants to leave since no one else knows their modules as thoroughly as they do. The speed was reduced dramatically in the end, which mostly was due to decreasing of motivation among the project members.

The **documentation** made during the project were project definition, requirements specification, component descriptions, object diagrams, sequence diagrams, test cases, user documentation, help files, and meeting protocols. The technical documentation was written by the programmers and the user documentation was written by the consultant department. X wrote the project documentation and the project manager and X also wrote some parts of the requirements specification when the consultant department was short of time.

6.3.4 Company 3

The company was founded in the late 90’s after a fusion of two divisions in a large telecom company and today they have roughly 900 employees spread over five cities. They are a consulting company focusing on telecommunication and their business is structured around four different divisions: Network-, Business-, Enterprise-, and Mobile Solutions. Their largest customer is the same company which they emerged from, in 90-95 % of all projects conducted their former employer has the customer role. This was also the case in the project covered in this interview.

The project was planned to last four months and covered an additional application to a web based sales promotion system. The system is owned by their customer, but it is administrated by Company 3 and they also perform all further development. This system has been around since the middle of the 90´s and since then a number of additional applications have been added. The purpose of this particular project was to develop an application for handling a new telecom product with supporting services, which had been developed by their customer. The project had a fixed price based on earlier projects, the price can be corrected later if necessary e.g. new requirements or unforeseen needs might be thought of by the customer during development. These changes are then handled in a specific routine and dialogs with the customer.

Person X is trained system developer and market economist, during the system developer education X used e.g. the programming languages Cobol and Pascal, but it was only at the very first job that X has actually worked as a software developer. The later jobs have instead included marketing, sales, and education in these areas. X was hired at Company 3 as a project leader, but today X also work as a system administrator for the sales promotion system. In the latter role X actually plays the role of the customer within the company, X has the responsibility to collect new requirements or requests from the customer and also estimate the time and cost for these together with the developers in the project. In this role and as a project leader X also mediates between the customer and Company 3 when issues that affect the project or the system as a whole are discussed. The customer has a person that holds a corresponding role as X does at Company 3, all communication between the two parties was handled by these two persons.

As already mentioned the reason for this project taking place was that their **customer** had developed a new telecom product that they needed support for in the sales promotion system. The project involved seven persons at Company 3, one was X as project leader but there were also four developers and two testers. The product developed was a three-tire application consisting of a presentation layer, an application logic layer, and finally a storage layer. The presentation layer was developed in Java, the logic in a C based tool called Tuxedo, and for storage they used an Oracle database and also a number of old systems used by their customer to which the logic layer was connected.

For their projects Company 3 uses two different software processes, one for fixed price projects and one for current account projects. The software **process** for fixed prices is sequential and consists of three main phases, these phases are analysis, development and finally the test phase. This was the software process used in the project we covered in our interview.

To be able to allocate the right resources for the project, the dates for when the different phases should be ready were established early on, these and other important events were used as milestones for the project. The initial time table had to be reworked already in the first phase due to additional requirements added by the customer, this lead to that the project was delayed one month. After this update of the time table, Company 3 did not need to correct it again but the project lasted five months instead of four as was planned from the beginning. Thus this did not cause too much trouble since it was performed so early, even before any development had begun.

The first thing that happened in this project was the **analysis phase** in which their customer contact delivered a first draft of a requirements specification. With this specification as a base Company 3 started their analysis phase, during this phase a product specification was developed for the project where the customer requirements were made clear and further defined. They always try to find and analyze all requirements during this phase, but in reality and also in this project requirements were appearing also during the development phase. If these late requirements should be added or not depends on the customer, since they own the system and also pays for the development they have a lot of power. However Company 3 must always first analyze and decide if the requirement is reasonable and possible to implement before the customer can decide if it should be included or not. The time estimations that were made in this project were based on experiences from earlier projects, it was X together with the developers that performed these estimations. The developers estimated the parts that they later developed since they are the ones with the most knowledge about theses parts. A very rough design was also developed in this first phase.

The rough design was then further developed in the **development phase** which followed the analysis phase. During this second phase all the functionality were planned to be developed, but there was also some development during the test phase due to errors found that had to be corrected and some misunderstandings concerning the requirements specification. Components tests of their own parts were performed by the developers during the development phase to assure that the components were behaving correctly in isolation. Company 3 uses a code standard for their development for making the structure etc. of the source code look the same all over, by doing this they have much easier to understand each others code.

When the development phase was performed they continued with the **test phase**, in this phase the testers carried out system tests to see if the different components also worked well together. The test cases were developed by the testers themselves and parts of these were later used as an acceptance test by the customer. The customer developed their own test cases from the system test specification and then ran tests with their own data. Before the final product was delivered it was handed over to the administration at Company 3. Since they are responsible for the administration of the entire system, they also performed some tests to assure that the new product worked correctly with the existing system. The project team also handed over all produced documents to the administration, they used checklists to make sure that nothing was overlooked. Exactly what was included in this delivery was agreed upon by the customer and Company 3, thus the product, all produced documents, and approved test protocols were at least delivered in this specific project. The project team was also responsible for informing and train all support personnel before the product was delivered. In this case the developed product was planned to run at one of the customer's sites. The administration had to report the planned delivery both in writing and orally, and then they agreed upon a date and time with the operation personnel at the customer's site for the delivery. The administration together with the project leader then performed the delivery after working hours since the main system was used during the days.

The **communication** within the project group was handled via discussions and e-mail. They had daily discussions throughout the entire project, this was possible since they were located very close to each other in the same building. Project meetings were held once every week with all the project members, the testers were also present from the beginning because then they could observe all changes performed during the project. Company 3 was convinced that this would help the testers to conduct more précis tests of the product and they also felt it important for them to start early with the test cases. The communication with the customer was also handled via discussions and e-mail. Progress meetings were held with the customer on a weekly basis. Since the customer knows Company 3 very well, their trust in them is very high. This is also due to the knowledge Company 3 has concerning the main system since they have developed it and also manage the administration. During these progress meetings the current state of the product was discussed and if there were any deviations from the plan these were immediately brought up for discussion.

The **project members** had great freedom, they were free to solve e.g. technical solutions as they whished so long as it did not affect the budget or time table negatively. All the developers had also the right to make changes in each others source code without informing their leader every time. The different parts of the product were assigned so that the developers with the knowledge in a certain area handled the parts related to their area of expertise. To prevent lost of knowledge due to a person leaving the company, they sometimes let less experienced personnel handle a certain part of a product to widen their knowledge. This is thus not performed in all their projects due to time shortage. To motivate their personnel Company 3 every now and then arrange for some activities, this could be laser game, dinner etc. All the members of this project had a number of years experience of the main system and similar development projects. Since they corrected the time table as early as in the first phase and they had based their estimations on experiences from earlier projects, there was no need for the project members to work overtime.

To control the **progress** of the project X followed up the time estimations to compare them against the time it really took to perform the activities. The completed activities were then compared against the time table to check that they were not falling behind the planned schedule. It was X that had the responsibility to perform these comparisons and also to have control of the progress as the project leader. The project members used a time reporting system to report the hours that they spent on the different activities. The CASE tools used in this project were SourceSafe for handling different versions of their source code and then they also used two tools which they have developed themselves. The first one was used for handling changes that occurred during the project and the second for designing the structure, components, interfaces between components etc. With the change handler tool they could later trace changes that had been made, e.g. who performed the change, where it was made, in which version it is located. All these different changes could then be found with help of SourceSafe where all versions were kept.

During this project Company 3 produced a number of documents, this **documentation** included different kinds of project management documents, product specification, test specifications, system specifications, user manuals, progress reports and also a final report. Besides these documents all developers also documented the source code that they had developed. It was the testers' responsibility to produce the test specifications and X handled the management documents, progress reports and also the final report. In this specific project the product specification was produced by a person outside the project group. It was this specification that the developers used throughout the development to assure that they developed exactly the product that the customer wanted.

The customer has demanded that Company 3 in the future must be more effective regarding their working procedures and costs. Due to this demand and that they also believe a new software process is needed, they have plans of introducing both the RUP and Ericsson's project management method (PROPS) in to their organization. Since their customer is so important for them, they have to do everything in their power to keep them especially now when the telecom market have been on the downgrade for a couple of years.

6.3.5 Company 4

The company was founded in the late 1980's and today they have roughly 30 employees in two cities, but are also represented by partners in a number of other countries. Their main focus lies on developing a mobile sales system that sellers can bring with them in the field. Their customers vary from small to large companies. The customer in this specific project was a rather large one.

The project covered in this interview lasted approximately 3 months and was a development project of their mobile sales system, which is a module based system. The **customer** bought the mobile sales system a while before they wanted an addition, which was a development of a discount reservation function. The mobile sales system is a standard system and a pre-study is made before every development project is started to check if it is possible to make the addition to the mobile sales system according to the requirements in the development project.

Person X has a corresponding bachelor exam in systems analysis, consisting of informatics and computer science. X has also studied the law and pedagogy. This is the first job in the IT business for X, but X has also worked with a great number of jobs in other businesses. X was employed as a software developer, but today X work as a technical project manager and X is also product responsible for their mobile sales system.

As already mentioned the reason for this project taking place was that their customer wanted to add a discount reservation function to the mobile sales system they bought from Company 4. The estimated amount of time for the project was 200 hours and involved four persons at Company 4, one developer for the module, one person for the parameter setting of the business logic, one person responsible for the print-out from the function developed, and X who was a project manager. The function developed was a three-tire application consisting of a presentation layer, an application logic layer, and finally a storage layer. The presentation layer was developed in VBA, the logic in C++, and for storage can either a Microsoft SQL database or a Sybase database be used.

All development projects at Company 4, which are of the type covered in this interview, use the same software **process**. This software process is divided in three different stages: requirements-, development-, and test stage. These stages can not be compared to phases since they do not have any transition criteria for progressing from one stage to the next. The reason for not dividing the software process in different phases is that it runs over such a short period of time. The software process follows a **requirements** specification template that should be filled in during the project. The first thing that happened in this project was that the customer came to Company 4 with a wish of a discount reservation function. X started a pre-study to specify their wish and to check if it was possible to make this addition to the mobile sales system according to this wish. The result of the pre-study was that it was possible to add this function as an additional module that can be chosen if a customer wants it or not. A consequence analyse was made to find everything that could affect the mobile sales system. This particular function did fulfil the requirements for adding it to the mobile sales system. A requirements specification was also the result from the pre-study, which was shown to the customer for acceptance. Everything in the requirements specification that was not approved by the customer had to be changed by Company 4. They always try to find and analyse all requirements during this stage, but in reality and also in this project requirements were appearing also later in the project. The planning of the project was made by checking for available resources, or in reality lack of resources. The amount of time needed from each resource was estimated and then X had to check if the required resources were available to do their tasks. It was X who made all the time planning for the project. X made the design of the module by trying to make the presentation layer as identical as possible to the rest of the mobile sales system and the database design by adding properties and sub objects to their own developed business logic layer. When all requirements were approved concerning the function proposal and the time plan was made, the **development** of the module could start and continued until all requirements were fulfilled.

The first **tests** that were made in this project were the system tests, which were followed by the business tests. The system tests were made by the developers and the business tests were made by X. Both of these tests were made by walking through the requirements specification but the business tests were performed by also looking at the consequence analyse. There were no deliveries or demonstrations before these tests were made. The customer made their own acceptance test without help from Company 4, since this would have made the customer to run the same tests as Company 4.

Much of the **communication** within the project group was handled via discussions but changes and corrections were handled in a product database. This database consisted of a list of changes and corrections, with attributes like prioritization, id, assigned person, and etc., which needed to be made. This product database spared them from misunderstandings, unnecessary double-work, and forgetting changes and corrections. A meeting was held every week where the events of the past and next week were brought up. These meetings were not pure project meetings since they also brought up things outside the scope of the projects. Discussions about problems and solutions to these problems were also brought up during these meetings. All project members participated in these meetings and sometimes also some other employees at Company 4 participated. A couple of meetings with the customer were held to identify the first requirements and to accept the final requirements specification. There was also a running communication with the customer throughout the entire project by e-mail and telephone. Company 4 did not use any consultant during this project.

The **project members** had to inquire about all major changes before they made them but minor changes and some artistic freedom were very appreciated in this project. All project members had the right to go in and change in other’s code if necessary and they have also tried to widen their knowledge to get a better understand of all parts in the system. This have unfortunate not worked and the most experienced person in an area is often the person that develops everything in that area. Company 4 do not have any kind of bonus system for these kinds of projects but they do have kick-offs for larger project sometimes.

The **progress** in this project was measured in hours and lack of time was of course compensated with more time. They did not use any milestones. The CASE-tool used was Microsoft SourceSafe. It was used for both the source code and some of the documents. Requirements changes were added to the product database and the requirements specification. The reason for adding the changes to the requirements specification was that they had to be able to go back and se what was changed. There was a constant dialogue with the customer, during these dialogs they told the customer about all changes made to the requirements specification. All these changes were able to be traced to the requirements and the source code was also able to be traced to the requirements.

The only **documentation** created for this project was the requirements specification and the meeting protocols. Any additions at the intern meeting were added to the requirements specification. Both design specification and test results were added to the requirements specification which resulted in only one document where all important information about the project were gathered. A user manual has been created for the entire mobile sales system but Company 4 did not write a new user manual for this module. Company 4 uses a code standard for their source code, by doing this they have much easier to understand each others code.

6.3.6 Company 5

The company was founded in the late 90’s and has today after a large restructuring roughly 10 employees. Their main focus lies on providing consultant services to small and medium sized companies.

The project covered in this interview was planned to last for approximately four months, during the fall of 2000. However the project continued for two to three months more with a lot of updates and added functionality that was not planed in the original project plan. The product developed was a web based email system with an import function to gather data from some other systems. They got the project through a public purchasing that lasted approximately 6 months. The **customer** was a rural district consisting of a project group with a project manager who handled the communication with Company 5. The project manager acted as an intermediary between the project group and Company 3. All thoughts and ideas from the project group went through the project manager.

Person X is trained software engineer and was employed at Company 5 in 1999 as a software developer. Today X current position is project manager, but development is still a large part of the job. It is X first employment in this line of business.

Company 5 have developed a software **process** that was followed by this project. This software process was in reality a couple of documents that should be created during the project. There was a very rough time planning made with only a few milestones, the finish of the requirements specification, the finish of the development and a pilot installation. It is the requirements-, development-, and the test phase that are the most important phases according to X and these phases should be marked with a clear take-off. The lifecycle of the project was sequential so the phases identified were performed one after another.

Company 5 won the purchase because they had an offer that included images of how they thought the layout should look like, no other company had thought of that. There where three requirements specifications made, one for each part of the system. It was Company 5 who prioritized all the requirements. They took approximately one week each to create but almost half of the system where programmed before they were approved. Company 5 had to withhold the customer a little concerning making up new requirements, this was requirements that was not in the offer from the beginning or made the user interface looking disordered. However, the requirements that did not generate too much work was carried out and added into the to the requirements specification without any further confirmations by the customer. At the end of the project, the final requirements specification was saved to be used as a base for when other customers requests a similar system in the future.

In the **requirements phase** Company 5 performed the analysis, design, and of course identified the requirements as the name reveal. However, in this specific project they also started to implement as early as in this phase. The **development phase** then continued the development of the system which was made in three parts with an interface to communicate with each other. The communication between the different parts of the system was made with XML. Some of the component tests were made during the development phase. All other tests were made in the **test phase** and these tests were performed on a development environment that was set up to simulate the real operation environment, these tests were component, integration, and system tests.

The pilot installation was made before the final delivery so the customer could test the system. The pilot phase lasted two weeks and in the end they had an evaluation meeting with the customer to discuss their new viewpoints concerning the system. Most of the viewpoints brought up by the customer were changed to their satisfaction. After this a new two weeks long test period was performed and one more evaluation meeting was held. At the end of this phase a full installation was made, the system should then run for a month without any disturbances before the customer wanted to accept the delivery. This, however, took rather long time because of some error corrections that had to be made. The acceptance agreement was made in consultation with the customer.

They had a project meeting once every week to discuss the progress of the project. Most of the **communication** was made orally but some of the project members were disturbed by all the chatter which lead to that they started to use email instead. The confidence between the customer and Company 5 was very important and they had a lot of meetings to discuss the system with the customer. X thinks that it is very important to look after their customers and be flexible when it comes to the requirements. They changed a lot of small things in the requirements specification that the customer did not think of in the beginning of the project. They met the customer approximately once every week during the implementation phase but much more often during the requirements-, test-, and pilot phase, almost daily. The customer visited them almost once a week, and if the project group felt that the system was in a presentable state they showed the new features or the user interface to them. Company 5 got help from consultants concerning the user interface and some implementation problems.

The project consisted of four **project members** and one 'helicopter' project manager who participated at project meetings once every week which lasted roughly 30 minutes. The 'helicopter' project manager was in charge for the time planning and the discussions during these project meetings, but he had no contact with the customer. X was the real project manager in this project and the other three project members were programmers. Everyone participated in creating the requirements specification but the design and the architecture were created by only one of the programmers. The programming languages used for the system was Active Server Pages (ASP) and VB. The database used was Microsoft SQL. Each developer was responsible for the part of the system that he/she had developed. The parts that each developer had developed should only be changed by them and no one else. The developers were placed on positions that match their knowledge and they had a lot of freedom concerning how technical problems should be solved. They had approximately 10 hours overtime per week during the development and testing phase in this project. For motivating their personnel Company 5 have always had some kind of bonus system, e.g. a weekend trip to celebrate a successful project.

The **progress** of the project was measured through comparing the criteria for each phase with the criteria in the time plan. The 'helicopter' project manager and the project manager kept track of the time and the progress in the project. The CASE tools used in this project were Microsoft SourceSafe for handling different version of all files, the user interface was made in Homesite, and Microsoft Visio was used in some parts of the design. The requirements specification was written in Microsoft Word. None of the requirements were able to be traced to the code or the test cases.

During this project Company 5 produced a number of documents, this **documentation** included different kinds of project management documents, product specification, test specifications, system specifications, user manuals, and also a progress reports. Besides these documents all developers also documented most of the source code that they had developed. The developers wrote the documents for their parts of the system and the project manager wrote all project documents. The software process that was followed in this project was based on earlier experiences.

6.4 Summary

In this chapter the summaries of the performed interviews were presented and also how and on which grounds we selected the software companies and projects for our investigation. At an early stage we had to reconsider our criteria for the selection of software companies since we realized that it would be extremely difficult to find the ones that we first wanted to interview. So instead of focusing on similar software companies we performed our interviews with software companies which differed in size and also in the type of software products they developed.

Some of the areas covered and discussed in these interviews were the software companies and the developed product, the software process used during the project and its phases and activities, how they communicated within and outside the project group, and what kind of documentation they produced during the project. The purpose with these areas and belonging questions was to establish how these software companies performed their software development, what their project lifecycle looked like, which activities they performed, and also more Agile related factors as how they dealt with communication, freedom for the developers etc.

7 ANALYSIS OF INTERVIEWS

7.1 Introduction

In this chapter the four lifecycle characteristics identified in the Agile software development chapter will be compared against the project lifecycles used by the interviewed software companies, this is done to establish the similarities and differences between them. In addition to these characteristics, we have identified two more Agile software development characteristics applied by these companies. Namely simplicity and handling of changing requirements late in the development cycle, these will also be analyzed in this chapter.

Our criteria for the comparison is that if any of these software companies fulfil at least our four first characteristics this can be seen as their lifecycles have very much in common with the Agile ones, however we do not require that these software companies should have exactly the same phases or that the characteristics should be performed exactly as described by XP or DSDM.

7.2 Types of projects

Since the first four interviews covers projects for updating an existing software product, some of the activities normally associated with software projects are not performed to the full. This specially concerns activities that are performed early in the project lifecycle, in XP it is the exploration phase that is the most affected one and in DSDM it is the feasibility- and business study phases. This is because the need for all the output that these phases generate is not as great as in projects for completely new products since the customers and their needs and wishes are usually fairly well understood. So to perform e.g. a regular business study for these customers could be seen as a waste of time. Thus this does not imply that Company 1, 2, 3, and 4 in this case would have performed the exploration phase according XP or the feasibility- and business study according to DSDM if these would have been projects for completely new software products.

Here follow short descriptions of the types of projects covered in our interviews, this is done to clarify the different starting points for these projects and also the fact that some software companies have different software processes for different kinds of projects.

The project performed by **Company 1** was an update of a system that is used by several of their customers. Similar updates are performed every four months to deal with new requirements and correction of possible faults, the frequency of the updates has been agreed upon with their customers. The lifecycle followed in all their update projects is sequential with three main phases, namely requirements-, development-, and test phase. In addition to the software process used in these update projects, Company 1 also has a software process for when they develop completely new software products.

Company 2 performed another kind of update project for a large system that is used by many of their customers. They updated the system completely to raise the technology and make it component based. Updates like this are performed approximately every second year, but this particular project lasted almost three years. The lifecycle followed in these kinds of projects are sequential and consists of three main phases, these are requirements-, development-, and test phase. During these large updates they regularly improve and update the previous version, the now running version of the system, to handle new requirements and different customer opinions.

The project that **Company 3** performed was an update project to add an application to one of their customers systems. This update was requested by the customer and was planned to last four months but due to changed requirements by the customer the project lasted five months instead. The lifecycle followed by Company 3 in these kinds of projects are sequential and it has three main phases, namely analysis-, development-, and test phase. Company 3 has two different types of software processes, one for fixed price projects and one for current account projects. This particular project covered in our interview was handled as a fixed price project.

Company 4 performed an update project for a specific customer, which added a module to their version of Company 4's mobile sale system. It was the customer who requested this update and it took Company 4 approximately three months to complete the project. The lifecycle followed during these kinds of projects are sequential but it does not have any clear phases identified, however Company 4 performs three stages during this software process: requirements-, developments-, and test stage.

Company 5 performed a project in which they developed a completely new product for their customer. They started the project after winning a public purchasing against other software companies. The project was planned to last four months but it continued two to three more months due to updates and added functionality which was not included in the original project plan. The lifecycle followed in this project and all other projects that Company 5 performs is sequential with three main phases identified, these are requirements-, development-, and test phase.

The fact that our interviews covers different kinds of projects, update projects and a project for new software development, affect our analysis since these are quite different and by this the final result can not be compared directly against each other. Additional aggravating circumstances are that the update projects differ in a couple of aspects and that most of these software companies apply different kinds of software processes for different kinds of projects.

The projects performed by Company 1, 2, 3, and 4 concerns different kinds of updates, these include updates for a single customer and for several customers, the projects differ in size, and the reason for the updates also differs, in one case it is for improving the technology whereas in another it is the customer that requested the update. Since we have concentrated on specific projects to receive more concrete information about these software companies´ project lifecycles, we have also limited the ability to see the whole picture of how these software companies really conduct their work since most of them use two different kinds of software processes. This fact became clear to us during this analysis, and due to these above mentioned factors we can not claim with absolute certainty that these software companies apply our identified lifecycle characteristics to the full or not. We are only able to express our conclusions about these characteristics based on the projects covered in our interviews. In this chapter we will also discuss some influential factors such as if an update project can be seen as a partial project included in a larger one and by this be an iteration in that larger project.

7.3 Iterative development

None of the interviewed software companies applied any iterative development in their projects, instead their software processes used a sequential lifecycle. However as mentioned in the previous section, the interviews with Company 1, 2, 3, and 4 covered projects for updating an existing software product. These projects could then be seen as small iterations in larger projects, except maybe for Company 2 which performed their update project for almost three years without showing or delivering anything during this period of time.

There are no clear similarities regarding the phases of the different lifecycles used by all of these software companies compared to XP which have six phases and which should be performed iteratively. The same is true for DSDM which also uses iterative development but it has five different main phases. The output from the iterations in XP and DSDM is used to give feedback for the following iterations and by this improve the system being developed.

The project conducted by **Company 1** used a software process which had a sequential lifecycle with three main phases. The valuable feedback usually collected when performing iterative development was also used by Company 1 in their own special way, despite their sequential lifecycle. The way Company 1 conducted their project support that sequential lifecycles work very well when they in turn are combined with other sequential lifecycles in an iterative fashion. This was possible since they have performed similar projects before that provided experience and good input to the current project, and also that there will be new projects after the current one that can take care of the requirements that were not implemented.

The lifecycle of the project conducted by **Company 2** was sequential, with three phases: requirements-, development-, and test phase. The project lasted for almost three years without showing or delivering anything until the test phase started after two and a half years. The contact with the customer, or as in this project the consultant department, during these phases varied a lot. They had a lot of meetings in the requirements phase, almost none in the development phase, and a vast number in the test phase. The absence of communication with the customer during the development phase was a disadvantage because the customer was surprised how things actually worked when they tested the system in the test phase. An iterative development lifecycle usually escape this problem by collecting feedback from the customer frequently.

The software process used by **Company 3** had a sequential lifecycle with three main phases identified. Even thus that they followed this sequence of phases during their project, minor deviations occurred. All requirements were planned to be identified and defined during the analysis phase, but some new requirement also appeared later on in the software process during the development phase. To approve these late identified requirements they had to perform an analysis just for these to be able to carry on with the development. Testing was another activity that was performed differently in comparison to the sequence in their software process lifecycle, the fact is that all developers tested their own parts as early as in the development phase. Despite these two deviations their development was still performed sequentially and not iteratively.

Company 4 used a software process that was not divided in different phases since it ran over such a short period of time. However, the lifecycle of the project was divided in three different stages: requirements-, development-, and test stage. These stages can not be compared to phases since they do not have any transition criteria for progressing from one stage to the next, in other words a sequential lifecycle was used. Feedback from the customer was collected by constant contact via e-mail and telephone.

Also **Company 5** used a software process that had a sequential lifecycle and due to this no iterative development was performed. However they did occasionally return to the requirements phase from the development phase to analyze requirements identified late by their customer, this was done to support the decision if the new requirement was to be included in the delivery or not. The developers at Company 5 also performed tests of their own parts during the development phase, but this did not affect the sequential flow of their lifecycle.

7.4 Active customer collaboration

XP and DSDM uses prototypes in addition to the standard communication channels, this is to let the customer see and try the product being developed at an early stage and then give feedback that helps to improve the product. XP also requires an on-site customer that can be available all the time to answer questions, they should also write their so called user stories and also choose the sequence of the iterations performed during the development. The importance of collaboration and cooperation between all stakeholders in a project is highlighted by DSDM and seen as very important.

Company 1 used active customer collaboration during their project and this is something that they always try to do. They handled their customer communication by frequent discussions via meetings, telephone, and e-mail, and this was done throughout the entire project. Their customers were able to affect which requirements to include in the update and also how the layout of the user interface should be designed in cooperation with the product responsible. Company 1 always include a number of different documents e.g. user manuals, information documents intended for their customers when the product is delivered, occasionally they also train their customers for managing the new product. Even though that Company 1 did not have exactly the same collaboration with the customers as described by XP or DSDM, they still had a very active one.

Active customer collaboration was used in the requirements phase and the test phase in **Company 2** but not in the development phase. They had approximately 10 meetings with the consultant department during the requirements phase and almost no contact, except discussions about new requirements, during the development phase. However, the contact with the consultant department was brought up again in the test phase since it was the consultant department who should make the tests of the system developed. They did not have any meetings with the real customers of the product being developed since all requirements came through the consultant department. Active customer collaboration was not fully used in this project since Company 2 did not communicate at all with the consultant department during the development phase.

Due to their history together **Company 3** and their customer have always had a very close and active collaboration and this was also the case during this project. The customer produced the first draft for a requirements specification and then they were also allowed to come with new requirements throughout the entire project. This was possible in this case since the customer owned the main system and the product being developed, and then of course they did pay for possible delays caused by requirements added late in the project. This also led to that they had great power and control over what happened in the project. The communication continued during the entire lifecycle of the project and was handled through discussions via progress meetings on a weekly basis, phone, and e-mail. Company 3 was also responsible for training the support personnel for the product since they were the ones with the best knowledge about the product. In addition to this they also helped their customer with writing a number of different user- and technical manuals for the developed product.

A couple of meetings between the customer and **Company 4** were held to identify the first requirements and to accept the final requirements specification. All questions were discussed with the customer as soon as they appeared. Company 4 did not use a customer on-site as described in XP but it was the customer who wrote the first requirements which could be compared to the request in XP that the customer should write the user stories. The frequent contact with their customer also indicates active customer collaboration.

Company 5 had a very active collaboration with their customer during the project, their communication was handled through frequent discussions throughout the entire project via in advance booked meetings, unannounced visits by the customer, phone, and e-mail. Something that Company 5 always strives for is to grant their customers wishes as much as possible and this implies that they need to be very flexible concerning all requirements identified in their projects. The customer in this project was able to affect which requirements that should be included and the design of the user interface of the system, this was done through frequent and open discussions with Company 5.

7.5 Frequent releases

This characteristic is related to the one about iterative development, due to their sequential lifecycles none of these software companies performed frequent releases during their projects. However as already mentioned the update projects can be seen as a small iteration of a larger project and then the delivery in these projects covered could be frequent releases within the larger project. Both XP and DSDM use frequent releases with short intervals to show the progress of the system to the customer on a regular basis, the customer can then give feedback which may be used to improve the system for the next release. DSDM advocates that releases should be delivered once every week up until a couple of weeks.

Due to the sequential lifecycle used by **Company 1** in their project, they did not perform frequent releases in this particular project. The project included only one single delivery at the end of the project when the system had been thoroughly tested. However Company 1 occasionally also distribute smaller updates via internet during their larger update projects like this one, but these are only delivered when Company 1 believes it is necessary and not on a regular basis. The fact that these larger update projects results in releases every four months supports that Company 1 really do perform frequent releases if these projects are seen as small iterations in a larger project, but since we only have knowledge about a specific update project we can only assume how the development is performed as a whole in such a project based on the above mentioned factor.

The first release in the project conducted by **Company 2** was made when the test phase started. Almost two and a half years after the project started. Thus no frequent releases were made in this project and no feedback from the customer was collected until the test phase started. However, feedback from the releases in the test phase was collected and used to improve the system for the next release.

Since the lifecycle in the software process used by **Company 3** was sequential and that they also applied it more or less according to its definition, no frequent releases were made during this project. The only release that took place was the final delivery of the product at the end of the project.

Company 4 had only one release of the module they developed, the final deliver at the end of the project. This was due to that the size of the project was very small and that the lifecycle of the project was sequential.

Two releases were performed by **Company 5** during this project, these releases were for the pilot testing and the final delivery. Since these were performed at the end of the project they can not be seen as frequent releases as defined by XP and DSDM. At most of the customer visits at Company 5, the system was shown for the customer but only in its current state. These small presentations were not planned in any way, the customer was just shown new features or the user interface of the system in its development environment. Due to the sequential lifecycle and the factors mentioned above, Company 5 did not perform frequent releases in their project.

7.6 Continuous testing

Both XP and DSDM stress the importance of testing throughout the entire project and not just at the end before the delivery. XP advocates that as many tests as possible should be automatic ones since they believe this is the only way to prove that a feature really is working as planned. In DSDM the testing is not seen as a separate activity in a specific phase, it should be performed in all phases and these tests are handled by the timebox mechanism.

Company 1 believes that continuous testing is extremely important and this was also performed during this project. However they do differ in how they carry out their tests compared to XP and DSDM. They did not perform any automatic test nor did they use any mechanism similar to timeboxes in DSDM to handle these tests in their project. All tests performed by Company 1 were performed manually by their developers or specific test personnel. They started to perform functional tests as early as in the development phase to assure themselves that the components developed would function correctly in isolation before they were integrated. Then they carried on to perform different kinds of tests e.g. integration-, and system tests until the system finally was delivered.

Testing is one of the foundations when developing software in **Company 2**. They have a well structured testing procedure but they do not use any of the methods in XP or DSDM. All functionality was tested by the developers, they performed component- and integration tests. The test phase in this project was made in four steps: free tests and tests of the new requirements, tests of the main functions in the system, detailed system tests, and tests of the graphical interface. The acceptance test, which was made by the consultant department, tested all requirements in the requirements specification. Thus, different kinds of tests were made continuous throughout the project which both XP and DSDM advocate.

Tests were performed by **Company 3** continuously throughout the entire project, this was done during both the development- and test phase. Developers performed component tests on the components they had developed to validate their correctness before integrating them. The testers then performed integration- and system tests for validating that the entire product worked according to the agreed upon requirements. Before the product was delivered the administration division also performed their tests to be sure that the product that they were going to be responsible for was working correctly. All tests performed by Company 3 were carried out manually no automated tests were used.

Company 4 tested throughout the entire project, as described in both XP and DSDM as very important. The developers performed system tests while developing the module, which was followed by business tests before the module was delivered to the customer. Company 4 did not use any kind of automatic tests during this project. The final acceptance test, which was made by the customer, was made without any help from Company 4.

Company 5 tested the system throughout the entire project, some component tests were performed as early as in the development phase, and during the test phase they performed component-, integration-, and system tests. This continuous testing was performed by Company 5 to assure them that the system worked as planned also in an early stage of the development and to get feedback on the developed parts. All of these tests were performed manually by the developers on a simulated operation environment. Before the final delivery Company 5 performed a pilot installation at the customer, this was done so that the customer and the intended users could try and test the system further.

7.7 Simplicity

There is a focus on simplicity in all Agile software process, this is also true for XP and DSDM. This includes both activities performed in these software processes and which documents that should be produce during a project. XP always produces as simple design as possible since they only are dealing with today’s problem and not what might happen in the future. Something that is true for both of these software processes is that they always try to do enough and no more, this minimalism lets them do only the necessary activities and produce only the documents that will provide value for the product or the project. This is also true for the interviewed software companies, they have realized that they in reality only uses three main phases for their software process, except for Company 4 which performs similar activities as the other software companies but they do not have any distinctive phases which these activities are performed in. The identified phases used by the other four software companies are requirements/analysis-, development-, and test phase, these are the only phases performed since these are seen as the most important ones and the only ones which a special criteria is needed for entering and leaving. In addition to this they only produce documents that they believe are needed during the project or by the intended users of the developed product. Due to these two factors we mean that all these software companies fulfil the simplicity characteristic described by Agile software development.

7.8 Changes of requirements

In Agile software development, welcoming of changing requirements even late during the development cycle is advocated to help the customer with their competitive advantage. In both XP and DSDM focus is on satisfying the customers and users of the system being developed and to accomplish this, new or changing requirements must be handled also late during the development cycle. This was also applied by all the software companies, despite that they did not perform any iterative development they still handled late requirements to please their customers.

7.9 Conclusions

The analysis of the interviews which we have conducted was meant to show the similarities and differences in a clear and understandable way between the project lifecycles used by the software companies and the software processes lifecycles described by XP and DSDM. To be able to do this we identified four lifecycle characteristics for XP and DSDM which we then compared against the lifecycles used by these software companies in the analysis, this was done to see if the software companies fulfilled these characteristics or not.

As the analysis carried on we realized that it would be very difficult to claim if these software companies did work as the Agile software processes describe or not, the fact that it depends on quite a few different things surprised us a little. In reality it is very difficult to decide if a software company work Agile like or not, since it depends on e.g. what kind of project it is, do the software company use different kinds of software processes for different kinds of projects and how are these structured, and if update projects are thought of as small iterations, was the entire system also developed iteratively from the beginning? Due to the above mentioned factors and that we focused on a single project that these software companies had performed made us very limited concerning our findings and conclusions, since we only could base these on the specific projects.

However we did find some similarities between the software processes lifecycles used by the software companies and the identified lifecycle characteristics of XP and DSDM. These were active customer collaboration which was applied by four of the five software companies and continuous testing which all thought of as very important and also applied during their projects. The way these activities were performed in these projects did not however correspond to the way described by XP or DSDM, and none of the projects followed a software process which had similar phases as described by the two Agile software processes. The characteristics iterative development and frequent releases were much more difficult to validate than the two mentioned above owing to the factors mentioned in the previous section. The fact is that sequences of update projects are quite iterative in its nature and often also release products frequently, for each update they improve existing features or add new ones. We had to bare this in mind when we analyzed the interviews to present as truthful information about these characteristics as possible, as already mentioned we analyzed all projects as separate ones and by this the update projects were also seen as sequential since no iterative development or releases were performed during their lifecycles.

In addition to the four identified lifecycle characteristics we did identified two other Agile software development characteristics during the analysis which were applied by all of the software companies. These companies were all handling requirements which were changed late during the development cycle and they were also applying simplicity to their projects. All of them performed only the activities and produced the documentation which they felt were really needed for the particular project and the software product.

The projects that we chose for this investigation has made the analysis much more complicated than we originally thought. All projects except one have been update projects and the comparison of these projects lifecycles and the lifecycles of XP and DSDM have stand apart quite considerably. The update projects have e.g. skipped the phase where the market is investigated. Projects that covered the lifecycle of a whole product would probably have been much more suited for a comparison with the lifecycles of XP and DSDM since both covers a whole product lifecycle, everything from the beginning with a market investigation to the end with no more requirements to develop.

Even thus that we did find a number of similarities and differences between the lifecycles used by these software companies and the ones described by XP and DSDM, we strongly believe that if we would have selected similar projects as mentioned above it would have been easier to draw even more precise conclusions from this investigation.

7.10 Summary

In this chapter the analysis of the interviews was presented, this was done by first describing the five projects briefly and discussing the fact that these projects were quite different that in turn lead to difficulties when we analyzed them. The problem was that the results from the analysis could not be compared directly against each other since these projects differed greatly in e.g. size and also that they were different types of projects, one was a project for new software development and the other four were update projects of an existing software product.

Then the four XP and DSDM lifecycle characteristics which we have identified were compared to how these software companies performed their software development in these specific projects. Our analysis showed that two of these characteristics, namely active customer collaboration and continuous testing were applied by all software companies during their projects, except for one who did not practice the first one of these. However, these software companies did apply the characteristics in their own way and not as described by XP or DSDM. The other two, iterative development and frequent releases were more difficult to decide if they were applied or not. This was because we only focused on a single project and that the update projects could be seen as small iterations in larger projects which also then could have frequent releases. The only thing we could establish concerning these two characteristics was that no iterative development or frequent releases were performed during the lifecycles of the projects we covered in our interviews. In addition to these conclusions we could not find any clear similarities between the lifecycles used by these software companies and the ones described by XP or DSDM concerning the different phases that should be performed. We did however identify two other Agile software development characteristics during the analysis that were applied by these software companies. These were simplicity and the ability to handle changes that occurred late in the development cycle.

8 DISCUSSION AND CONCLUSIONS

8.1 Introduction

In this chapter we present three suggestions for further work based on the work we have done for this master thesis. A discussion is also held about our hypothesis and where to place the software processes used by the software companies in relation to the Agile software processes. Finally we present our general conclusions about the findings identified in this thesis.

8.2 Suggestions for further work

We have investigated five projects in five different software companies and compared these with four identified characteristic with focus on the lifecycles of XP and DSDM, and two more Agile general characteristics. This comparison has been very difficult to make and there are a lot of areas that still needs to be investigated.

Suggestion one: Some parts of the projects were comparable with the six identified characteristics. E.g. active customer collaboration was applied by four of the five software companies and continuous testing was applied by all of them. The way these activities were applied in these projects did not however correspond to the way described by XP or DSDM. Other parts like iterative development and frequent releases were not applied by any of the software companies. We think that it would be interesting to do an investigation that tries to implement the activities that were not applied by the interviewed software companies to a software company that resembles any of these software companies. Would this increase or decrease the time, cost or functionality for one of their projects?

Suggestion two: Some of the characteristics were applied by all software companies and some were not applied by any of the software companies. Are there any reasons for why some of these were not applied by these software companies? This master thesis could be seen as a base for an investigation that finds out more about this.

Suggestion three: Another similarity that we found between these software companies was that a majority of them used three different phases: requirements-/analyse-, development-, and test phase. We have not made any analysis of why they only used three phases. It would be interesting to see the reasons for why some similar software companies only uses three phases and not six phases like XP or five as DSDM.

8.3 Discussion

Our hypothesis for this master thesis was that many of the ideas in the Agile manifesto and its twelve principles were common sense and also used by software companies which not explicitly claim that they are using an Agile software process. However our investigation has shown that these ideas might not be common sense for all individuals and companies in the software industry. Four of our six identified characteristics were performed by most of the software companies, but two of the most distinctive ones within Agile software development were not performed by any of these software companies. Agile software development in general and also XP and DSDM stress the importance of iterative development and frequent releases for producing a successful software product. Despite these ideas all of the interviewed software companies performed their development sequentially with one large delivery at the end of the project. This fact made us wonder if all ideas behind the Agile manifesto and its principles really are common sense as many intercessors of Agile claim, or is this just true for certain individuals and some software companies. If these are common sense why are they not followed by all software companies?

All of the software companies in this investigation followed sequential waterfall like software processes, between the different phases in these software processes they also had predefined milestones to know when one phase was finished and that it was suitable to enter the next one. However the latter was only practiced by four of the five software companies, since one of them had a very short project they did not have any distinctive phases or milestones. Despite this difference their software process was very similar compared to the ones used by the other software companies.

To visually give an overview of these software processes in relation to Agile software processes as XP and DSDM, Figure 4-2-1 ‘The planning spectrum’ can be used. In this figure the Agile software processes are placed on the left hand side of the spectrum where more flexible and adaptable development types are located according to the originator of the figure. Our view of the software processes used by the five software companies in these specific projects are that all of these should be placed on the right hand side between the middle of the spectrum and the ‘Milestone plan driven models’. This is due to the fact that all of these software companies used plan driven software processes and that four of them also were milestone driven. Even thus that the software companies in this investigation were quite flexible e.g. when their customers wanted to change or add requirements, this however can not be compared to the flexibility and adaptability practiced by the Agile software development processes. By performing iterative development with frequent releases the Agile software development processes facilitate for much faster and more frequent changes to occur during the software development. According to us it is very difficult to just place a software process on a spectrum as described above and place it in a specific category since many different factors might affect how a software process really is used. However the best software process according to us is the one which is fully adjusted to fit a specific project and its characteristics, instead of having a general software process for all projects.

8.4 General conclusions

In this master thesis we have performed a comparison between the lifecycles used in projects for software development by five different software companies which explicitly claim that they do not use any Agile software process and the ones described by the Agile software processes XP and DSDM. The reason for choosing this topic was that after examining the Agile manifesto and its principles we felt that a number of the discussed issues in these were common sense to us and as it later turn out also to many intercessors of Agile, and that these also must be applied by many non-Agile software companies. The lifecycle slant was chosen since it covers many aspects of software development and also to limit the size of the topic to a reasonable level for our master thesis.

Before performing the interviews with the five software companies we conducted a literature survey, this was done to investigate and get a better understanding of the software process lifecycle model, Agile software development and specially the Agile manifesto and its principles, XP, and DSDM, and also for looking deeper into the investigation approach that we should use for collecting the most valuable data for our thesis. The above mentioned areas are described in the thesis to make these clear to the reader and make it easier to understand the interviews and our analysis of them.

The interviews we performed were done with a project manager at each of the software companies since these persons have great knowledge about the software process used and its activities. We focused these interviews on specific projects that the software companies had performed recently to be able to get more concrete data of how the software development really was performed instead of how they should perform it according to some documented software process. To handle the questions and areas covered in the interviews and the analysis of them, we used a qualitative method since this is a very useful method when dealing with ‘real life’ aspects of a specific topic. For the analysis we identified four lifecycle characteristics which are true for XP and DSDM, most of these are also true for the Agile manifesto and its principles. Two Agile characteristics were also included in the analysis, these were identified while performing the analysis of the other characteristics. These were then compared against how the five software companies performed their software development in relation to these characteristics. After the analysis we presented our conclusions regarding the results generated from the analysis.

Our hypothesis that much of the Agile manifesto and its twelve principles are common sense and used by even non-Agile software companies was shown by this investigation to be incorrect. As could be seen in the analysis, we did not look into all the aspects of the manifesto and principles in full. Our concern was the lifecycles of XP and DSDM, however most of the characteristics we identified and used during our analysis are part of the manifesto and its principles. Two very important characteristics according to us were not applied by any of the software companies in these specific projects, namely iterative development and frequent releases, instead all followed a waterfall like software process. The other identified characteristics were not fully applied either by the software companies in this investigation. This fact made us understand that the issues discussed in the Agile manifesto and its principles are not so obvious to all software companies as we first thought.

However the software companies did not strictly follow a waterfall like software process nor can they be seen as following an Agile software development like process, they have instead adapted the software process to fit their needs. This together with the fact that these software companies did handle late changes to the requirements if their customer requested them to do so, correspond very well to one of our initial thoughts that these software companies do have a prescriptive software process which they try to follow but they can diverge from this if it is necessary. With this investigation we want to show that the ideas behind the Agile manifesto and its principles might not be as much common sense as we first thought and many intercessors of Agile claim. Another purpose of this investigation is to present how five different software companies really work during their software projects, and we feel that a truthful picture of this has been presented in our master thesis. The most sensational with their way of working is the fact that all of them are using sequential waterfall like software processes which by many has been condemned as old and very inflexible.

9 REFERENCES

Bailey, P., Ashworth, N., and Wallace, N. (2002), *'Challenges for Stakeholders in Adopting XP'*, XP2002, pp. 86-89, available from Internet (http://www.xp2002.org/prog_full.html) (13 March 2003)

Barrow, Patrick D.M. and Mayhey, Pam J. (2000), *'Investigating principles of stakeholder evaluation in a modern IS development approach'*, The Journal of Systems and Software 52, pp. 95-103

Beck, K. (1999), *'Extreme Programming explained'*, Addison-Wesley, ISBN: 0-201-61641-6

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jefferies, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001), *'Manifesto for Agile Software Development'*, available from Internet (http://www.agilemanifesto.org) (3 March 2003)

Boehm, B. W. (1988), *'A Spiral Model of Software Development and Enhancement'*, IEEE Software, pp. 61-72 (30 March 2003)

Boehm, B. W. (2002), *'Get ready for Agile methods: with Care'*, IEEE Software, pp. 64-69

Boehm, B.W. (1989), *'Software Risk Management'*, IEEE Computer Society Press, Los Alamitos, ISBN: 0-8186-8906-4 (case)

Boehm. B.W., Egyed, A., Kwan, J., Port, D., Madachy, R. and Shah, A. (1998), *'Using the WinWin Spiral Model: A Case Study',* IEEE Software, pp. 33-44 (30 March 2003)

Cockburn, A. (2002), *'Agile software development'*, Addison-Wesley, Boston, ISBN: 0-201-69969-9

Charette, R. (2001), *'The decision is in: Agile versus heavy methodologies'*, Cutter Consortium Vol. 2. No 19, pp. 1-3

Davis, A., Bersoff, E. and Comer, E. (1988), *'A Strategy for Comparing Alternative Software Development Life Cycles Models'*, IEEE Software, pp. 1453-1461

DSDM Consortium, *'Delivering Agile Business Solutions On Time'*, available from Internet (http://www.dsdm.org) (15 Mars 2003)

Fowler, M. (2000), *'The New Methodology',* available from Internet (http://www.martinfowler.com/articles/newMethodology.html) (18 Feb 2003)

Fowler, M. (2002), *'The Agile Manifesto: where it came from and where it may go'*, available from Internet (http://martinfowler.com/articles/agileStory.html) (17 Feb 2003)

Fowler, M. and Highsmith, J. (2001), *'The Agile Manifesto'*, Software Development, available from Internet (http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm) (26 Feb 2003)

Highsmith, J., Sutherland, J., Glass, L. R., Moss, T. L., Kruchten, P., Wagner, L. and Russell, L. (2001), *'The Great Methodologies Debate: Part 1'*, Cutter Consortium Vol. 14. No 12, pp. 1-44

Holme, I. M. and Solvang, B. K. (1986), *'Forskningsmetodik Om kvalitativa och kvantitativa metoder'*, Second Edition, Studentlitteratur, ISBN: 91-44-00211-4

Howard, A. (1997), *'A new RAD-based approach to commercial information systems development: the dynamic system development method'*, Industrial Management & Data Systems 97/5, MCB University Press, pp. 175-177

Humphrey, W. S. (1989), *'Managing the Software Process'*, Addison-Wesley, Reprinted with corrections, ISBN: 0-201-18095-2

Jacobson, I., Booch, G. and Rumbaugh J (1999), *'The Unified Software Development Process'*, Addison-Wesley, ISBN: 0-201-57169-2

Kvale, S. (1997), '*Den kvalitativa forskningsintervjun'*, Studentlitteratur, Lund, ISBN: 91-44-00185-1

Maurer, F. and Martel, S. (2002), *'Extreme Programming Rapid development for Web-Based Applications'*, IEEE Software, pp. 86-90

McCauley, R. (2001), *'Agile Development Methods Poised to Upset Status Quo'*, SIGCSE Bulletin Vol. 33. No. 4, pp. 14-15

McCormick, M. (2001), *'Programming extremism'*, ACM Vol. 44. No. 6, pp. 109-111

Microsoft Corporation (1999), *'Analyzing Requirements and Defining Solution Architectures MCSD Training Kit'*, ISBN: 0-7356-0854-7, available from Internet (Sample chapter) (http://www.microsoft.com/mspress/books/WW/sampchap/3869.asp) (3 March 2003)

Miles, M. B., Huberman, M. (1994), *'Qualitative data analysis: an expanded sourcebook'*, Sage Publications, Thousand Oaks, California, ISBN: 0-8039-4653-8

Nawrocki, J. and Wojciechowski, A. (2001), *'Experimental Evaluation of Pair Programming'*, pp. 1-8, available from Internet (http://www.pairprogramming.com) (17 Dec 2002)

Orr, K. (2002), *'CMM Versus Agile Development: Religious Wars and Software Development'*, Cutter Consortium Vol. 3. No 7, pp. 1-29

Patel, R., Tebelius, U. (1987), *'Grundbok i forskningsmetodik'*, Studentlitteratur, Lund, ISBN: 91-44-24851-2

Patton, M. Q. (1980), *'Qualitative Evaluation and Research Methods'*, Second Edition, Sage Publications, Newbury Park, California, ISBN: 0-8039-3779-2

Royce, W. (1998), '*Software Project Management, A Unified Framework'*, Addison-Wesley, ISBN: 0-201-30958-0

San Diego Mesa College, *'CIS 210: Systems Analysis and Design'*, available from Internet (http://teachers.sdmesa.sdccd.cc.ca.us/~gmerx/CISC210-SystemsAnalysis/CIS210ClassSessions.htm) (20 February 2003)

Sommerville, I. (1982), *'Software Engineering'*, Sixth Edition, Addison-Wesley, ISBN: 0-201-39815-X

Sommerville, I. (1996), *'Software Process'*, Computing Department, Lancaster University, Lancaster, UK, available from Internet (http://www.dimap.ufrn.br/~jair/ES/artigos/sommerville.pdf) (24 Feb 2003)

Stapleton, J. (1997), *'DSDM: business focussed development'*, Second Edition, Pearson Education Limited, ISBN: 0-321-11224-5

Svenning, C. (1996), *'Metodboken Samhällsvetenskaplig metod och metodutveckling'*, Third Edition, Lorentz förlag, ISBN: 91-972961-3-9

Trost, J. (1993), *'Kvalitativa intervjuer'*, Studentlitteratur, Lund, ISBN: 91-44-39401-2

Turk, D., France, R. and Rumpe, B. (2002), *'Limitations of Agile Software Processes'*, XP 2002, pp. 43-46, available from Internet (http://www.xp2002.org/prog_full.html) (29 March 2003)

Warsta, J., Salo, O., Ronkainen, J. and Abrahamsson, P. (2002), *'Agile software development methods. Review and analysis'*, VTT Electronic Espoo, pp. 1-107

Wells, D. (1999), *'When should Extreme Programming be Used?'*, available from Internet (http://www.extremeprogramming.org/when.html) (12 March 2003)

10 APPENDIXES

10.1 Terms and Abbreviations

Term	Explanation
DSDM	Dynamic Systems Development Method
XP	Extreme Programming
PROPS	Ericsson's project management method
RUP	Rational Unified Process
MSF	Microsoft Solution Foundation
SCRUM	SCRUM Software Development Process
Crystal	Crystal methodologies
UML	Unified Modeling Language
ASD	Adaptive Software Development
FDD	Feature Driven Development
IT	Information Technology
RAD	Rapid Application Development
VB	Visual Basic
VBA	Visual Basic for Applications
ASP	Active Server Pages
SQL	Structured Query Language
CMM	Capability Maturity Model
BTH	Blekinge Institute of Technology

10.2 Interview guideline used in the survey

- The interviewee
    - Experiences
        - Prior roles
        - Education
    - Role/Roles in the project

- The project
    - Characteristics
        - Type of software product
        - Type of customer
        - Reason for starting the project
        - Programming language
        - Code standard
        - Number of persons involved
        - Length of the project
    - Roles
        - Rotation of roles
    - Lifecycle
        - The beginning of the project
            - How were the requirements collected?
            - Were all requirements defined at once?
            - Planning
        - Was the project divided in different phases?

            - Analysis
            - How was the design performed?
            - How was the implementation carried out?
            - How and when was testing performed?
            - Integration
            - Criteria for completion of phases
            - Demonstration of the product to the customer
        - Sequential/Iterative approach

- Communication
    - Within the project group
        - How was the communication conducted?
    - Externally
        - How often did you communicate with the customer?
        - Were any consultants used during the project?

- Developers
    - Possibilities to make own decisions
    - Responsibility for the source code
    - Right person at the right place
    - Motivation

- Progress of project
    - Measuring of progress
    - Use of received information
    - Role

- CASE-tools
    - Use of CASE-tools

- Change management
    - How were changes handled?
        - Tracing
        - Was any kind of version handling system used?
    - Time consumption

- Documentation
    - Types of documentation
    - Time consumption
    - Roles

- Deliveries
    - Course of action at deliveries
        - Part deliveries or one large final delivery?
    - In which phase was the first delivery?
    - How was the delivery accepted?

- Process improvements
    - Handling of process improvements
        - When was this done?
    - Time consumption