Master Thesis
Computer Science
Thesis no: MCS-2008-40
January 2009

Are we ready for

Agile Development?

Daniel Barke

Department of
Interaction and System Design
School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Daniel Barke
E-mail: daniel.barke@gmail.com

University advisor:
Guohua Bai
Department of Interaction and System Design

Department of
Interaction and System Design
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 102 45

ABSTRACT

In the rapidly changing market of today, companies need to be responsive and react quickly to changes in both their competitors' behaviour but also to changes in their own technical environment.
In this thesis I have examined the agile characteristics of a number of companies in Stockholm, with focus on three agile concepts; Scrum, eXtreme Programming and Test Driven Development.
The work started off by a prestudy, in which I have identified the criteria that a company needs to fulfil in order to be considered agile. This resulted in four main categories of characteristics; Quality, Flexibility, Communication and Competence.
After doing the prestudy, these characteristics were investigated through a combination of a quantitative study and a case study.
While the results mostly lean towards agile behaviour rather than non agile, it was shown that a lot of work still remains, for instance regarding improvements in the communications area, and also in the way these companies apply the agile methodologies examined.

Keywords: Agile development, quality, flexibility, communication.

PREFACE

I would like to take the opportunity to thank my family for having such patience with me while I have been working on this thesis. Without your support, this work would never have been possible.

Thanks to my supervisor, Professor Guohua Bai, for being quick and responsive while coaching me, and for providing me with a lot of great feedback and important pointers.

I would also like to thank the participants of my quantitative study for taking the time to answer my questions and providing me with information invaluable for this work. Your input made up the core of this thesis.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Problem background

As a developer of the traditional schools, you would probably first write your application, with not much testing involved during this phase. Then, when the application is completed, you would either test it yourself or send it to a team of testers, and finally you would correct any eventual bugs and flaws that are found during those tests. A common approach used is the Waterfall model in which everything is done in a sequential order; Requirements, Design, Implementation, Testing and finally Maintenance.
However, developing software in this way, assuming that everything done in the previous steps was done to perfection, can be risky since to guarantee that would be as good as impossible.
Practicing such linear methodologies also limits a company's flexibility to market changes, since the project and the resources involved are locked up in long cycles. This way of working conflicts with the way the market works today.
A previous study performed on roughly 1000 IT projects points out waterfall practices as the single largest contributing factor for failure in no less than 82% of the cases [19].
Another study performed on over 400 projects applying waterfall model principles, with an average cycle length of six months, showed that only 10% of the developed code was actually deployed, and out of that only 20% was used [20]. Now, that is not very efficient.

Over the last decade voices have been raised, questioning these traditional methods, and out of this, various agile methods have grown. One of the more recent methods, still a hot topic, is Test Driven Development, TDD.

Developing software according to TDD principles means rethinking what you know and putting test and quality assurance before coding.
According to TDD, at first you should write a test case covering the functionality that you are about to add to the application. Naturally, this test will initially fail since the actual code is not written yet. When the test is constructed you should write the code that will make the application pass that test (and of course any previously written tests). This is done in small iterations, guaranteeing that the application is built in a controlled manner, where every section of code is tested before adding a new section.
Studies have shown that working like this not only improves the quality of written code, but it also tends to make the developer more productive [1].
Other studies have proven that iterative development is correlated with lower failure rates [19].

Besides TDD, I will also look at two other important concepts, namely Scrum, which is an iterative and very agile way of working, and eXtreme Programming, XP, which in short takes the best concepts of many practices and pushes them a bit further than usual.

1.2 Main problem

Introducing agile development practices in an organization seems because of several different reasons to be hard to achieve.
Are companies in general not mature for this new way of working?
What factors can help enable an easier adaptation process towards becoming more agile?

1.3 Purpose

The aim of this study is to provide a better understanding of what is necessary of a company to be able to start developing software according to Scrum, XP and TDD, and thereby become more agile and less vulnerable to change.
I believe that companies of today in general still are not nearly as prepared for and willing to change as they should be to be considered agile, and I hope by doing this research to be able to verify this hypothesis.

1.4 Scope

To perform a large scale study and a cross comparison of different regions within Sweden and rest of the world would be very interesting, but due to time limitations this cannot be done in the scope of this master thesis.

Instead, I have chosen to concentrate the study to larger private companies in the Stockholm area.

2 FRAME OF REFERENCE

"Uncertainty is inherent and inevitable in software projects and processes" [14]

In this chapter, I will present what I have learned by studying existing literature and articles on the relevant topics, and establish the base of knowledge that I will use for producing my analysis model.

2.1 The waterfall model

Intuitively, when you plan a task at work, an activity with your friends or simply your day off work you'd probably want things to happen in a certain order. This just seems to be the way that we humans work. We need structure and order (well, most of us anyway).
This has also been the general approach when it comes to software development. A way of working known as The waterfall model and variations of the same has been the most widely used.

The waterfall model is a sequential development process, where the work is divided into logical steps that are supposed to be performed in a certain order. The illustration below shows the most common way of presenting the process.

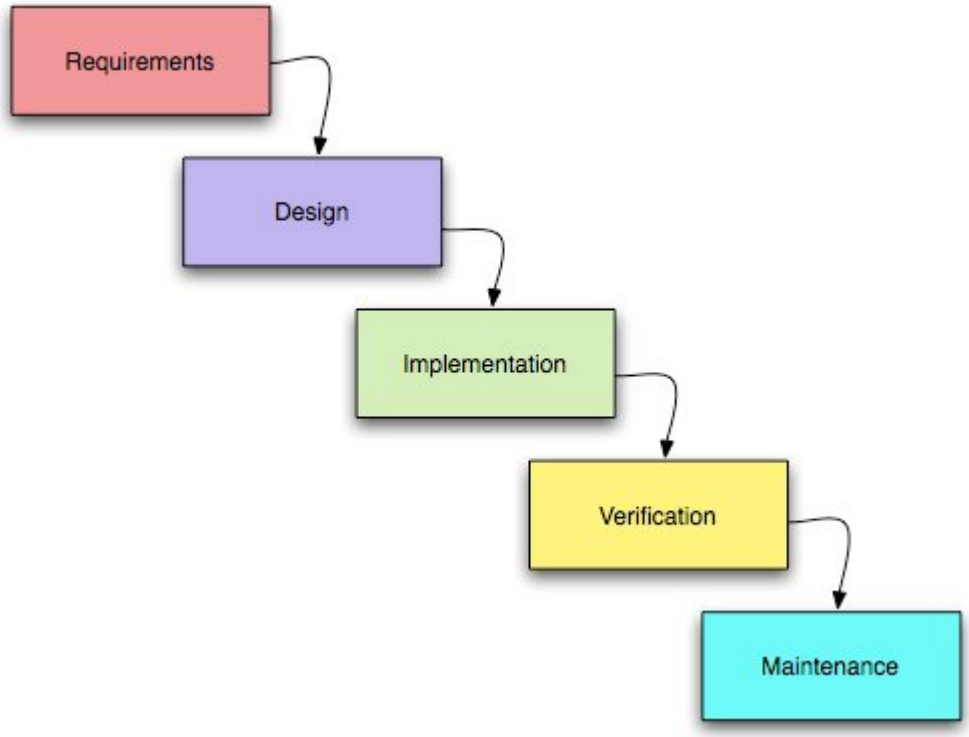


Figure 2-1 - The waterfall model, source: Wikipedia [26]

The basic principle of the waterfall model is that everything has to be done to completion in the first step before you move on to the second step, and so on. Likewise, you can't go back to a previously completed step to make any adjustments.

2.1.1 Discussing the waterfall model

Advocators of the waterfall model emphasize on the benefits of having a solid ground to stand on; Requirements analysis done to perfection before designing the application; Flawless design covering every possible scenario before starting to code and so on.

However, suppose you have completed the requirements phase, and the design is set, and you are way in on the third stage, developing the software that has been so studiously designed, when your client suddenly comes up with a new feature that simply has to be included in the final solution. How do you solve this? Well, you can't – at least not if you work according to the waterfall model.
Or, suppose the development phase is finished, and the application has been delivered to testing, where, naturally, some bugs are discovered. Well, you say, you just go back and correct the bugs, that must be allowed to do, right? - Yes, of course, but it just so happens that this specific bug originates not in the source code produced during implementation but in inadequate requirements. How should you solve this? Well, you cannot go back and change the requirements this late in the process, since that would conflict with the ways of the waterfall model.

These examples might not be very concrete; but they fulfill their purpose by demonstrating what has been found to be a huge weakness in the waterfall model, namely lack of flexibility.
Studies have shown that a typical software project experiences a 25% change in requirements [15], and not being able to respond to those requirement changes efficiently is a huge drawback.

2.2 Just-In-Time

"Having the right part at the right place in the right amount at the right time."

The foundation for today's agile methodologies of software development was actually laid in the car industry.

In 1922, Henry Ford, in his My Life and Work, presented, for the time being, a completely new way of thinking regarding stock levels:

"We have found in buying materials that it is not worth while to buy for other than immediate needs. We buy only enough to fit into the plan of production, taking into consideration the state of transportation at the time. If transportation were perfect and an even flow of materials could be assured, it would not be necessary to carry any stock whatsoever. The carloads of raw materials would arrive on schedule and in the planned order and amounts, and go from the railway cars into production. That would save a great deal of money, for it would give a very rapid turnover and thus decrease the amount of money tied up in materials. With bad transportation one has to carry larger stocks."[2]

This technique of keeping stock levels at a minimum and letting the current customer demand control the inflow of new stock is today known as Just-In-Time, JIT.

2.2.1 JIT in the software development process

Now, what if Ford's thinking was applied to software development?
Consider isolated pieces of functionality as your inventory, maybe not even implemented functionality, but even more important - functionality that still only exists at the requirements level. Wouldn't it be convenient to only take in new such inventory just when it is actually needed? This way of working would give great benefits, for example:

- Resources (developers) will not be locked up in very long cycles.
- The overall development process flow will become much more flexible and less vulnerable to change.
- The client, who is responsible for the requirements, and the development team will establish a much closer and lively relationship, which in the long run will have a positive result on the finished product.

2.3 Lean production

While Ford was the founder of the concept of JIT, Toyota embraced this way of thinking and developed it into their own model for working more efficiently.

Toyota started out as a rather small family company in the textile business and has today taken place as the world's most valuable car manufacturer [3].
A key factor for the company's success is their production philosophy known as Toyota Production System, TPS.
TPS takes the concept of JIT and adds to it the importance of elimination of waste, or non-value-adding activities, where seven specific types of such activities are defined, namely [4]:

- Overproduction
- Waiting
- Transporting
- Inappropriate processing
- Unnecessary inventory
- Unnecessary / excess motion
- Defects

2.3.1 Lean software development

While the TPS defines the philosophy in a very Toyota specific way, the concept has been generalized into what is known as Lean Production, or Lean Manufacturing (often mentioned simply as Lean), which can be applied to all types of industries.
In *Lean software development* [6], three major wastes have been defined, namely:

- Extra features - maps to overproduction in TPS
- Requirements churn - maps to unnecessary inventory in TPS
- Organizational boundaries - could be compared with Inappropriate processing

It is quite obvious that all three of the above issues are unwanted and that they can become huge obstacles to the development process. However, the occurrences of them are still way too common.
In order to have an efficient and flexible business companies really need to address these problems.

2.4 Agile development

"We are uncovering better ways of developing software by doing it and helping others do it" - Introduction to the agile manifesto [5]

In the fast moving market of today, where the prerequisites can change drastically from one day to another in virtually any type of industry, a company has everything to gain by not having their resources all locked up in long cycles and by not following a too detailed plan for the future. Now, that is not the same as saying that you shouldn't plan tasks in advance, but you shouldn't do it in a way that makes it hard to change your path if a sudden need for such a change should arise.
That is what agile development is all about – working structured, but in a way that still allows you to be able to quickly adapt and respond to sudden changes.
Goldman, Nagel and Preiss put it like this [17]:

"Agility […] is about succeeding and about winning: about succeeding in emerging competitive arenas, and about winning profits, market share, and customers in the very center of the competitive storms many companies now fear"

There are many methods of working agile, but I have chosen to study the two most widely used methods of today [18]; Scrum and XP, and finally a third concept; Test Driven Development, TDD, which has recently caught my eye, showing great potential.

2.4.1 Scrum

Figure 2-2 –Burndown chart, Source: Scrum and XP from the Trenches, Kniberg [7]

A key factor for being agile when developing software is to work in relatively small iterations, instead of one single linear flow, like the waterfall model would suggest.
A popular way of doing this and a process coming on strongly is Scrum.
There are complete books written about scrum and many variations in the usage, but I will try to explain the main concept shortly here.

First of all, the development team should together with the product owner decide on a proper length of their iterations. Three to four weeks cycles are commonly used. This is really a give-and-take situation, where the team generally benefits from somewhat longer sprints, to gain momentum, while the product owner often wants shorter sprints, to be able to follow and influence the work in shorter cycles [7].
The work to be done originates from a *product backlog*, in short a list of features that the product owner wants included in the final solution. Each task should be a relatively isolated feature that can be delivered and tested without depending too much on another feature. The tasks in the product backlog should also be prioritized by the product owner.
The person leading the team is called a *scrum master*.

Before each sprint, a sprint planning meeting takes place, where the team sits down and selects the tasks from the product backlog that they think they can fit in to and deliver during the upcoming sprint. Often the product owner will take place in these meetings as well, where he can reprioritize among the remaining tasks.
The selected tasks are moved from the product backlog into the sprint backlog, which then serves as the target list for the upcoming sprint.

Every day during the sprint, a *Daily scrum* should take place, where all team members can lift any eventually arisen problems to the group, and the scrum master can make sure that all team members stay on track, and know what they're supposed to do.
These meetings should be very short and are not intended for deep discussions on specific issues, but more to keep the whole team updated and leveled with each other, and for the scrum master to always have an overview of the current status. Another important thing with these daily meetings is that they should be open to anyone interested, for instance the product owner, customers or maybe other scrum teams, simply anyone interested in the team's progress.
After each iteration, the team should demonstrate their progress on a *sprint review meeting*, a rather informal forum that just as the daily scrums should be open to anyone interested.

To keep track of the sprint progress, a diagram called *Burn down chart* is used. In short, this diagram shows the estimated amount of work left on the y-axis, and the dates on the x-axis. See the picture leading this chapter for an example. This chart is an important tool in being open about the team's progress, since it should be visible to anyone interested, allowing anyone to quickly get an indication of how well the sprint actually is moving on [7].

2.4.1.1 Testing according to scrum

Process testing, i.e. testing complete flows of an application, has usually been done after a large delivery, by a team of testers somewhat isolated from the development team.
Scrum challenges this way of working and instead invites the testers to the team, to work completely in parallel with the developers. This gives great benefits in feedback and communication, which in the end results in better testing and thereby higher quality of the application.

2.4.1.2 Benefits of scrum

The main advantage of scrum is the iterative way of working in combination with a close relation to the product owner.
Assuming your sprints are chosen to a proper length your team is always flexible, and able to respond to new requirements.
Assuming proper time estimates are made, the team will always deliver something in the end of each sprint that can be demonstrated to the customer.

2.4.2 XP

Extreme Programming, XP, is an agile methodology that very well lives up to its name by aiming to enhance customer satisfaction through taking traditional software development practices to extreme levels.

XP rests on four core values [8]:

- Communication
- Simplicity
- Testing (Sometimes generalized as Feedback)
- Courage

Communication aims to make sure that everyone in the team has the same view on the requirements, and secondly that this view actually corresponds with the one of the users and the customers.
Simplicity can easiest be described as the aim to only code what is needed at the moment, and nothing extra. You shouldn't have to wait for the grand design documents that are to describe the complete solution, since these will probably change with time anyway.
The **feedback** value of XP can and should be interpreted in several ways. Of course feedback within the team, as well as from customers to team members is important. However, you should also let the actual code give you feedback. This might sound strange, but this is achieved through the habit of writing *unit tests*, and frequently running these to make sure your new code hasn't broken anything.
The last core value, **courage**, simply means to encourage the developers to dare to try to improve and *refactor* their code at any time. The courage should further be strengthened by the use of unit testing, since by having unit tests for every piece of code, you can always quickly tell if your refactorization left the functionality intact.

2.4.2.1 Practicing XP

How should XP be put into practice then? Some of the "extreme" rules are [9]:

- "*The customer is always available*"
  This assures that the expertise is always there for the team if any questions should arise. It is also valuable from the customer's perspective, since he also can get his hands dirty and influence the development in real time.
- *"All code is pair programmed"*
  By practicing pair programming, you're not only improving the quality of the produced code, but you're also making sure knowledge is distributed around the team, especially since you should also frequently rearrange the pairs.
- *"All code must have unit tests"*

This is an important rule that cannot be broken, since that would ruin the whole idea. But if you stick to it, and have well covering unit tests for all written code, you will have full control of your application. By making sure all unit tests pass before committing new code to the common code base of the team you also remove unnecessary bumps in the road, where one pair of programmers suddenly has to go in an correct bugs produced by another pair.

Apart from these rules, most of the thoughts behind scrum are adopted by XP, including well planned sprints and daily stand-up meetings.

2.4.3 TDD

"Clean code that works"

Traditionally, you'd write your code, and eventually finish the application, before it is put to the test. When working like this, testing is only used as a tool to verify the code. Working like this has its downsides. For instance, the tests will risk being influenced by the code, and at worst they might even be faked, written just to make certain code pass even though that code might not be correct [22].
Test driven development, TDD completely flips this way of working and suggests that you should instead let the test cases drive the production of new code.
TDD applies an iterative, evidence-based process to coding [22]. Now what does that mean?

2.4.3.1 Red/green/refactor

As a developer working test driven, when you're about to add new code, you will always start off by writing a unit test. This test is supposed to verify the functionality that you're about to add to the application, and only this functionality, nothing else. After writing the test, you should run it, just to verify that it fails – **red**.
The next step is to do the actual implementation, adding the code that is intended to make the test pass. An important note here is that you should only write just enough code to make the test pass, and nothing more. In fact, you are allowed to break any programming rules you've learned, just to quickly make the test pass.
Should you feel like writing more code than necessary, perhaps to extend the functionality, you need to create another test first. No exceptions to this rule.
After writing the code, run the test again to verify that it passes – **green**. Now, you might end up with the test still failing, and that is ok. Then you will just go back to coding until you solve the problem and make the test pass.
The third and maybe the not so obvious step is **refactorization**. This is allowed only after making a test pass. When all your tests pass, and you thereby know that the code works as intended, it is not only ok, but even encouraged to go back and refactor the code you've just written, for instance making it better structured, removing duplication, or making up for the rules you've disobeyed in the second step.
Otherwise, if you would skip this third step, you would probably end up with pretty nasty looking code, since the rules of the second step states that you should only write the code that makes the test pass.

These three steps conclude the whole idea of TDD - to work in small iterations, with constant control over whether your code works or not.

An interesting side note regarding refactorization; If you're not using the test first approach, and instead write your tests after implementing new functionality, you will often need to refactor the written code only to achieve better testability. However, since you don't have any tests verifying that code yet, you will have no way of knowing if your refactorization kept the code intact, which makes it a real nasty catch-22 [22].

2.4.3.2 Benefits of TDD

Kent Beck [10] states:
"Test-driven development is a way of managing fear during programming"

By working test-driven, as a developer, you can always be confident in your code, you will always know that it works since it is constantly tested. The code can be refactorized at any point with a secure foundation of regression tests [22].
Studies [1] have shown that working test driven also increases the productivity of a developer, so TDD is actually a technique that improves both quality and quantity at the same time.

Larman [16] points out another important benefit of practicing TDD:
"It is at least a semi-enjoyable way to do testing – that makes it more sustainable […] Traditional ("test last") testing is avoided because it is boring or tedious. By writing the tests first, the developer is engaged in thinking through the proper public behavior of the class not yet written. That's interesting and creative. And, she writes the test and then builds something to make it pass. That gives a small feeling of accomplishment."

In other words, practicing TDD not only forces writing the tests that otherwise would risk being left unwritten; it also tends to make the testing more meaningful to the developer. Johnson [22] puts it like this:
"TDD means that developers have constant feedback about progress, in the form of more and more passing tests demonstrating required functionality. This is very satisfying."

3 PROBLEM DEFINITION/GOALS

3.1 Being Agile

The process of adapting to agile philosophies and starting to work test driven and practicing Scrum and XP is not a simple task that can be expected to be done painlessly over night. Instead it has to be implemented in the whole organization.
In this chapter I intend to explain why this is hard to achieve, and go through the main criteria that need to be fulfilled in order to successfully become agile.

Peter Schuh has put together the following table as a summary of the differences between being agile and being non-agile [23]:

Project Environment		Project Characteristic	
Category	**Variable**	**Agile**	**Non-Agile**
The Development Team	**Communication Style**	Regular Collaboration	Only When Necessary
	Location	Collocated	Distributed
	Size	Up to 50 People	More than 50 People
	Continuous Learning	Embraced	Discouraged
Project Management	**Management Culture**	Responsive	Command and Control
	Team Participation	Manditory	Unwelcome
	Planning	Continuous	Up Front
	Feedback Mechanisms	Several	Not Available
The Customer	**Involvement**	Throughout the Project	During Analysis Phase
	Availability	Easily Accessible	Hard to Reach
Processes and Tools	**Team Input**	Team Has the Last Word	Team is Told What to Use
	Amount	Just Enough	More Than Enough
	Adaptability	May be Changed	May not be Changed
The Contract	**Requirements and Dates**	Flexible	Fixed
	Cost	Time and Materials	Fixed

Figure 3-1: Differences between being agile and non-agile.

Most of these characteristics can be found in the combined definitions of Scrum, XP and TDD, and therefore, this table gives a good hint to which factors I should investigate.

To make this even clearer though, and to really show the how well Schuh's table suits my work, I have taken the table and modified it by adding a column that states for each of the variable in which out of the definitions of Scrum, XP and TDD that that specific variables' agile characteristic can be found. I also removed the Non-Agile column, since those characteristics are of no interest in this study.

Project environment		Project characteristic	Found in which methodologies
Category	**Variable**	**Agile**	
The development team	**Communication style**	Regular collaboration	Scrum, XP
	Location	Collocated	Scrum, XP
	Size	Up to 50 people	Scrum, XP
	Continuous learning	Embraced	XP
Project management	**Management culture**	Responsive	
	Team participation	Mandatory	Scrum
	Planning	Continuous	TDD, XP
	Feedback mechanisms	Several	Scrum, XP, TDD
The customer	**Involvement**	Throughout the project	Scrum, XP
	Availability	Easily accessible	Scrum, XP
Processes and tools	**Team input**	Team has the last word	
	Amount	Just enough	TDD
	Adaptability	May be changed	TDD
The contract	**Requirements and dates**	Flexible	TDD, XP
	Cost	Time and materials	

Figure 3-2: Modification of Schuh's table, with column added for methodologies

3.2 Categorizing agility

In order to form some sort of structure around the questionnaire, and to be able to analyze the different aspects of being agile, I have divided the criteria into four main categories that sum up the agile characteristics that I will investigate.
These categories are

- Competence
- Quality
- Flexibility
- Communication

In the following chapter I will go through these, and elaborate on the questions that need to be answered in order to investigate the agile characteristics of a company.

3.2.1 Competence

In order to take on a new methodology, whether it is TDD or any other way of working, of course you have to learn the methodology first.
Education takes time, and lost time is to a company equal to lost money, so the company has to be willing to take such a cost.
Furthermore, not only the market is agile in the constant development of new products, but also the techniques and tools used by the developers change rapidly. Due to this non constant environment, a company must invest in education regularly to keep the competence up to speed with the market. Education can not be seen as a one time cost, it needs to be implemented in the company's general strategy.
How do companies of today look at this? Do they realize the importance of continuous education?
Where does the decision making regarding technical issues lie? Is it found on developer level, where it naturally should be because of the developers' technical expertise or is it found on management level just because the management "should" decide on everything?

3.2.2 Quality

Having identified education as a key factor for successfully becoming agile, the issue of cost also has to be brought up. In general higher quality can lead to higher costs and vice versa. You could also say that a product of higher quality could be more valuable to the customer, and can thereby bring in more profit to the company. Therefore I will investigate the relationship between these two factors.
As stated earlier, developing software test driven as opposed to developing linearly like according to the waterfall model, improves quality, and thereby in the end TDD can also raise a company's profit.
However, while this profit is only seen over a longer time, the initial investment of implementing agile methodologies made up out of mainly education and perhaps even recruitment will be noticed immediately.
How well can the companies appreciate the value of this investment? How quickly do they need to see the return on investment, ROI?
Developing software in a test driven manner assures improved quality of the delivered code. On the other hand, you could argue that constantly writing test cases consumes a lot of time, which would increase time-to-market, TTM, for the final product.
The question is though, is a product of poor quality, delivered on time really to prefer over a high quality product that has required a little bit longer time to develop?
How do the companies investigated in the study think regarding this deliberation? How quality oriented are they?
Another aspect is overproduction. TDD clearly states that no extra features should be developed unless they really are needed. From personal experience I know that it is not uncommon that a solution that looks simple on paper can become both more complex than it has to be and also contain a lot of unnecessary functionality. Do the developers examined have this in mind and try to avoid overproduction or do they tend to try to "show off" by producing more than what the customers' requirements state?

3.2.3 Flexibility

If the developers in a team have been working successfully in a certain way for a long time, and someone suddenly should come up to them telling them to forget everything they've learned in their career regarding methodologies and processes, asking them to totally revise their ways of looking at software development, that someone would probably have quite a hard time.
In order to achieve the adaptation, the developers have to possess the ability to change their ways for a better cause. They have to have the courage to try something new. Is this the case with developers of today, or can this be identified as a problem? Does age have an impact here?
In a larger perspective, looking at the whole company; how flexible is the company itself? Maybe the developers realize the benefits of Scrum, XP and TDD, but they end up obstructed by a management that lacks visions.
Another possible issue to take into consideration is that the company might have invested a lot in other methodologies and does not want to waste that investment by changing direction just like that. Is that a common thing?
What strategies are used regarding team sizes? A smaller team is naturally more flexible and less vulnerable to change than a larger team. Do the companies take this into consideration?
Looking at planning, specifically regarding requirements and design of a product, how flexible are the companies at this point? Is focus put on upfront design or is it seen as a continuous work throughout the development phase?

3.2.4 Communication

Both XP and Scrum put high value in communication. In XP this is for instance achieved through pair programming.
In Scrum, communication is the core of the whole process. Daily meetings, customers and testers working closely together with the developers, openness to the surrounding world through a public burn down chart and so on.
How well do the companies communicate within their own organizations and with their closest collaborators?
Do they actively promote and work on improving their channels of communication?
Are the customers always available for discussing any eventual issues during the development phase?
Are the companies using feedback mechanisms from the agile world such as burn down charts and automated tests?
What about strategies for locating the team members? Does the company put any effort in placing the team members together or are the developers distributed over different sites?

3.3 Cultural aspects

Apart from the main prerequisites stated above, it would also be interesting to study the effects of certain cultural aspects. I have identified four such external aspects and will here shortly state which, and reason around why and in what ways they might be of interest.

3.3.1 Size

Even though my survey will be sent out to companies of a certain size, where size is simply measured by the number of employees, the span of the selection can still be large enough to enable looking at differences between smaller and larger companies within the selection. Therefore, I shuold investigate the following; will a company's size have any impact on being agile? Will perhaps a larger company have a harder time adapting because of a more complex organization?

3.3.2 Geography

The survey will be sent out to companies in Stockholm. However, these companies do not necessarily have to be run mainly from Stockholm, or even from Sweden. What differences can be found between Swedish and international companies?

3.3.3 Degree of education

The need for continuous education on the agile concepts is obvious, but what about the background of the employees? I intend to get a fairly good picture of the educational background of the employees of each company and investigate if this factor can have any impact on the result.

3.3.4 Age

I personally have the belief that younger employees will make better way for using agile thinking in a company. By gaining a decent picture of the average ages of the employees of each company, I intend to see if this hypothesis can be verified.

3.4 Analysis model

In order to get a good overview of the elements that I should examine in my survey, I have produced a graph that visually represents the interesting factors.

The four main categories of factors presented earlier, Quality, Communication, Competence and Flexibility are related to each other in certain ways. For instance quality is a result of being flexible and having good competence. Further, in order to be flexible and to improve competence, good communication is necessary.

Taking these relations into consideration my analysis model looks as follows:

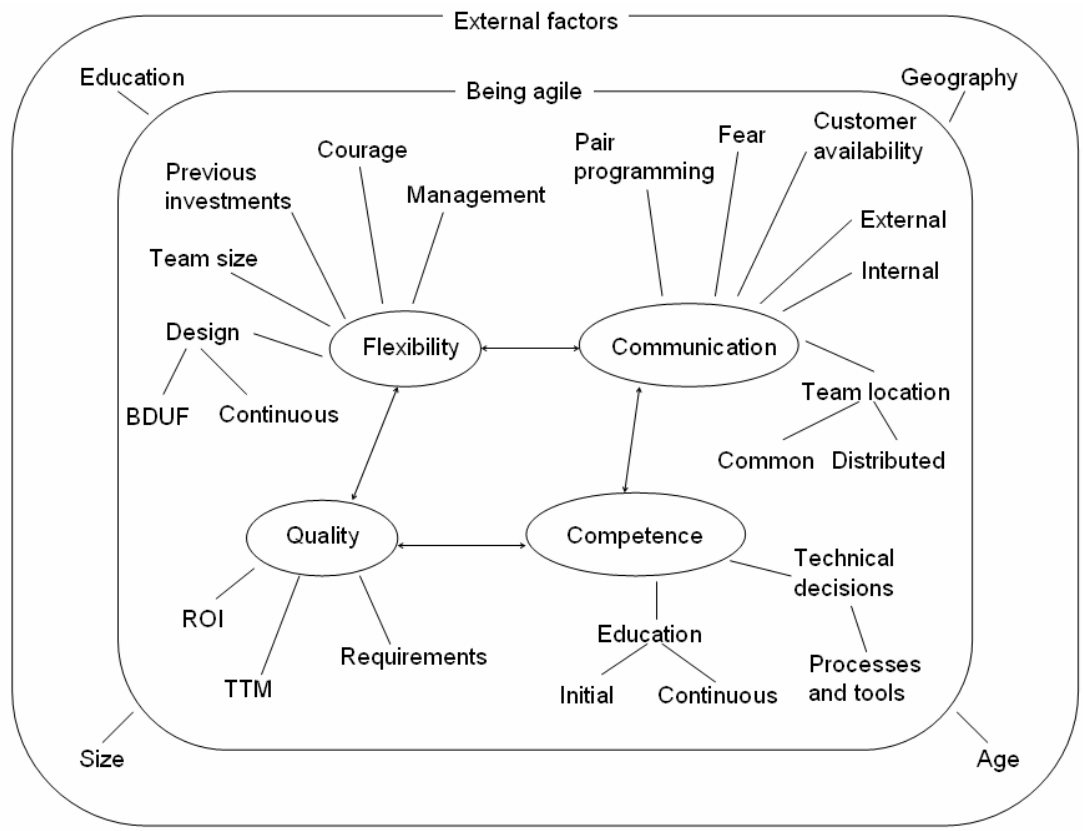


Figure 3-3: My analysis model for the survey, source: own

4 METHODOLOGY

In this chapter, I will present how I have been working during the composition of this thesis. First, I will state which methods I have used, and why.
After that, the ways of collecting data will be described, both for the frame of reference and for the empirical study.
Since I have performed a survey on a few companies to get an overview, I will explain how the survey has been put together and how the selection of participants was done.
Following that, I will explain how the work proceeded with a case study of a single company.
Then, I will discuss the ethical aspects considered, and finally a reliability discussion will end this chapter.

4.1 Prestudy

During the prestudy, I have studied agile ways of working, with the intention of producing a frame of reference that states the prerequisites for being an agile company. Based on that prestudy, I have composed a survey, which will be used for collecting data from a number of companies.
My sources of information have been books borrowed from the town library and the university library, e-books acquired on the web and articles found through well known research databases like Ebrary and Compendex.

4.2 Data collection method

When it came to the decision on which method I should use to collect data for the study, I essentially chose between two methods; either doing a survey study or doing a case study. Both of these methods come with their own benefits and drawbacks.

4.2.1 Survey study

A survey study can be performed mainly in three ways [12]; through face to face interviews, through telephone interviews, and through questionnaires, either sent by mail or posted through a website.
By doing a survey study through a questionnaire, I will be able to examine a larger group of objects, and I will have the possibility to present the results in charts and graphs. I will however not be able to go very deep on the topic, since I will not have the possibility to follow up an answer with another question based on that specific answer.
A survey is a quantitative study, and assuming the selection is made in a correct manner, a result that represents the whole population can be acquired by examining only a sample of the total population. [11]

4.2.2 Case study

A case study enables studying a specific phenomenon on an isolated object, i.e. an individual, a company etc, or a smaller group of such objects.
During such a study I would be able to go deeper into specific areas by using interviews. However I would not be able to generalize the results as easily as through a survey, since by examining only a few objects of the population I can not be sure that these results represent the whole of the population. I could end up with a result from examining one company that would verify my thesis, whereas examining another company easily might reject that same hypothesis.

4.2.3 My choice

Due to the time limitations on doing this research, I've decided to do a combination of a survey study and a case study.
My first intent was to only perform a larger survey, covering large companies in the Stockholm region, but realizing how time consuming that would be, and that it would need too much work to get a statistically valid result, I instead chose to do a combination of a quantitative and a qualitative study,
First, I will still do a survey, but in a much smaller scale than I originally intended, and with no intent of achieving statistical reliability, just in order to get a preview of what results to expect. Then I will do a single case study.

I decided to aim at getting representatives of ten large companies to answer my questionnaire, and then do a deeper case study on one of those ten companies.
By using a questionnaire, compared to doing interviews, I will get the most uniformed answers, and it will thereby be easier to compare the individual results.
Another benefit of using a questionnaire is that I will be able to use detailed questions with long answer alternatives.
However, I will need to carefully formulate the questions, and I also aim to test the questions in advance, since once the questionnaire has been sent out, I will have no way of clarifying or rephrasing my questions.
There is also always the risk of questions being interpreted in different ways by different participants [13], therefore I will have to try to minimize this risk by formulating the questions as clearly as possible.

4.3 Survey target group

I have chosen to study larger private companies in the Stockholm area. The reason for choosing Stockholm is that it is the capital of Sweden and houses the majority of IT companies in Sweden. To get a good statistical selection I have chosen to use the services of Statistiska Centralbyrån, SCB.
SCB categorizes all companies in a number of criteria [24] out of which three were of interest to me, namely type of business, region, and size, the latter measured in number of employees.
In order to get a reasonably sized group to examine, I have decided to base my selection on companies within the Stockholm county, in the computer programming business and with more than 100 employees. This resulted in a list of 132 companies.
Out of these 132 companies, I made a list consisting of roughly 15 companies to make contact with, with the intent of getting ten of those to participate in the survey.

After a full working day on the phone, calling switchboards and trying to get in touch with the correct departments and the right people to speak to, the result was a positive answer from five companies which I accepted as a satisfying result. The negative responses were evenly distributed between three reasons; Either the company policy was to not take part in surveys, secondly some companies had no actual programming operations in Sweden and therefore did not belong to my selection despite SCB's data, and thirdly I simply couldn't get in touch with the right persons on some of the companies. There were actually a couple of cases when I was promised to be contacted by someone that same day, but that never happened.
I also need to take into consideration the fact that the result of the phone calls to each company was very dependent on the mood and situation of the person that took my call. That person could for instance have a bad day or have a lot on his mind at the moment and would therefore not respond with as much enthusiasm and will to assist me as if he was in a good mood. This is clearly a random factor that cannot easily be avoided.

4.4 Survey composition

When composing a questionnaire, there are three main ways of asking the questions regarding type of answer; open questions, questions with fixed alternatives and questions where the answer is plotted on a scale [13].
In order to be able to easily compare the results and handle the answers mathematically I have chosen to use fixed alternatives.
The answers will be formulated in two ways, some according to the Likert-scale [25], which offers alternatives like "I strongly agree", "I agree", "I neither agree nor disagree", "I disagree" and "I strongly disagree", and the other questions will have concrete alternatives to choose from.
In order to increase reliability, I will use negations in a way that "I completely disagree" will be the most positive answer in some questions, placed randomly throughout the questionnaire. This way, the participants really have to read and understand the questions before answering them.

To achieve as high validity as possible, all questions will be based on the factors identified in my analysis model. Each question will have a purpose connected to at least one of the factors pointed out.

4.5 Ethical aspects

When doing research by examining people there are two ways of looking at the individual. Either, you see the research object as a part of the whole population, where this individual only is a tool used to understand the behavior of the whole population. On the other hand, if you emphasize on that specific individual, you need to consider the rights of that individual. These rights include protecting the personal integrity and also informing about the goals of the research. The individual must be allowed to decide on his/her own which type of information he/she is willing to give away.
Discretion and confidence are two important concepts, where the individuals being examined might demand anonymity and professional secrecy.
As a scientist you need to be aware of the responsibility you have towards the individuals you intend to examine [21].
Because of this, I will promise all participants in my study full anonymity.

4.6 Reliability

When performing a study where information is collected and processed there might be problems regarding reliability and validity in the information. Especially quantitative studies are sensitive for the reliability of the results. However, this problem can come up also in a qualitative study [21].

A discussion on the reliability is important, since the purpose of all research is to produce durable and valid results. When theories and models are being transferred to empirical observations, validity and reliability are important concepts.

To increase the reliability of my survey, and to inform the target companies in advance, I will before sending out the questionnaire call each of them to tell them about the purpose and the contents, ask them politely if they are willing to take the time for it, and to make sure the material will be sent to the right person.
Along with the questionnaire, I will send a letter where I shortly present myself and my goals of performing the survey.

The correctness of the questions will be established by testing the questionnaire in advance through letting a couple of professionals fill it out. This way I will be able to get input on unclear formulations and I will be given a chance to correct these issues before the survey reaches the participants.

When doing an interview or a performing a survey through a questionnaire there is always a risk that the participants might misunderstand a question [12]. Another risk is that the participants for some reason might tend to pick the answer that they interpret as the correct one instead of the answer that represents them.
This can not be completely avoided, but to minimize the risk, I will ensure full secrecy and anonymity, both during the initial contact over phone and in the letter sent together with the questionnaire.
This way, the individuals and companies participating in the study will be well aware of my aim to keep them anonymous, and I believe the results will be as justified as possible.
I will also offer the participants the possibility of contacting me over phone or mail if any questions would come up.

An important note regarding the results of the questionnaire is that even though I will present the answers as representative for a whole company, they actually only come from one individual who has been chosen to represent that company. That individual's opinions cannot for sure be said to match the opinion of the whole company.
The way to overcome this issue would be to let a large group of employees of the same company take the questionnaire and based on that get an average opinion, but that will be out of scope for this thesis.

5 EMPIRICAL STUDY

This chapter contains a presentation of my observations.
First, I will go through the survey by presenting and analyzing the results. The actual results of the survey mostly consist of numbers, and to make those numbers more meaningful to the reader, I have chosen to analyze the data while presenting them.
After that, the case study, done at a specific company chosen from the survey respondents, is presented.

5.1 The survey

The survey was sent out to five companies, with a deadline set ten days ahead. Out of these five, four returned their answers on time, and the fifth shortly after the deadline, so the return rate on the questionnaire was 100%.
In this section, I will present my findings from the survey split up into the four main criteria defined earlier.
The questions can be divided into five categories, where the same question can belong to more than one question in some cases. This division is shown in figure 5.1.

Background	1, 2, 3, 4, 5, 20, 21, 22, 24, 30
Competence	3, 5, 6, 13, 14, 17, 19, 23, 35, 38
Quality	10-16, 18, 27-29, 33, 39
Flexibility	6-17, 27, 28, 31, 33, 34, 36
Communication	6, 8, 23-29, 32, 36-40

Figure 5-1: Categorization of survey questions

Apart from the background questions, all questions ultimately intend to measure the agility of the respondent. To make it easier to compare individual results, I will translate the result of each question to a percentage, where a higher percentage always means a higher degree of agility. All (non-background) questions have either four or five options to choose between, therefore, I will translate these options to either 0%, 33%, 66% and 100% or, in the case of five options 0%, 25%, 50%, 75% and 100%.
As a consequence of this, I will interpret results over 65% as good, and results above 75% as very good in terms of being agile.
The complete table of results can be found in the appendix section.

Each of the four chapters will include charts of the results of all questions corresponding to the category of that chapter.

5.1.1 Competence

5.1.1.1 Empirical data and analysis

The most obvious question regarding competence is how high the share of staff that has a university degree is. Out of the participants, all answered that more than half of their staff has a university degree, and three of them even said that at least three quarters of their staff have a degree.
Next was the question of continuous education, where all participants said that they educate their staff at least one week a year and three of them even at least two weeks a year.

In order to have the best usage of the already existing competence at the company, the developers should interact much, and share their knowledge with each other.
Question number 6 investigates this, whether the companies actively promotes the developers possibilities to share knowledge and experience by not locking them up in single projects. The results were distributed between 25% and 75%, with an average result of 55%. I see this as a clear area of possible improvement.

Question number 19 was about the average working experience among the developers, and apart from one participant leaving the question blank for some reason; all others picked the same alternative, 5 to 10 years. A possibility here is that I might have made the span too big, but still this answer tells me that none of the companies in the survey have a high rate of "fresh out of school" developers in their teams but rather more experienced ones.

Question number 35 was about on which level the technical decisions lie in the company, and having one company not answering the question at all, three chose the alternative *"On developer level, with final approval of management"* whereas one participant stated that in their company, the management take these decisions with some influence from concerned developers. This rhymes badly with the definition of competence in my analysis model. However, still three of the respondents chose an agile alternative.
This question is closely related to question number 23, regarding choice of methodologies and tools, where the answers were distributed between the three top alternatives, averaging 70%. The respondent who chose the non agile alternative in question 35 also chose a non agile alternative on this question.
On question number 38, regarding the managements' perceptivity on suggestions and ideas for improvement from the staff, four participants answered *"well"*, and the fifth one chose the middle alternative, meaning neither good nor bad. This was actually the same company pointed out in the previous two questions analyzed. The average result on that question was 70%.

The average score on the competence questions in total was 62%.

5.1.1.2 Graphical presentation of the results

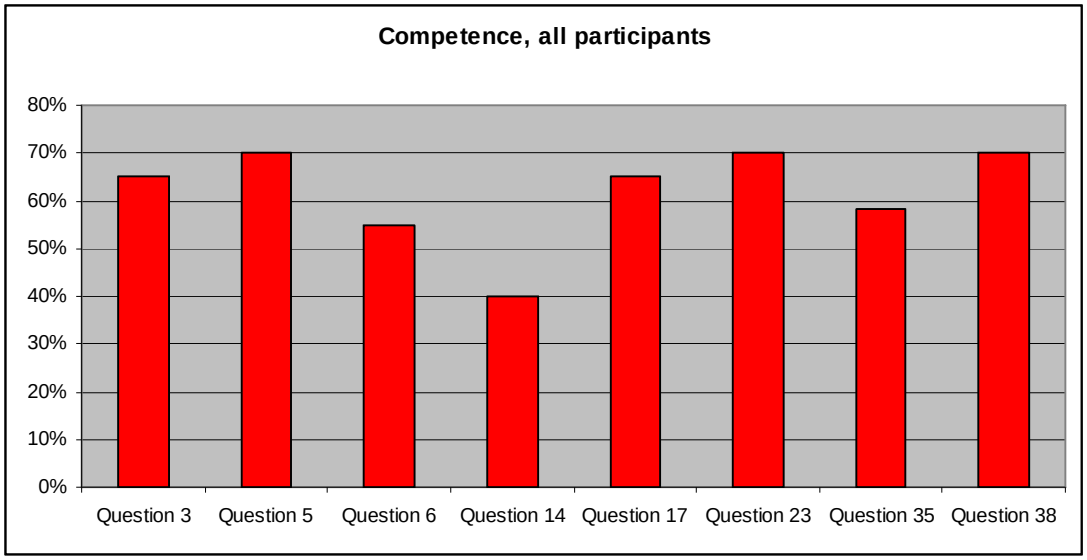


Figure 5-2: Survey results, competence, all participants

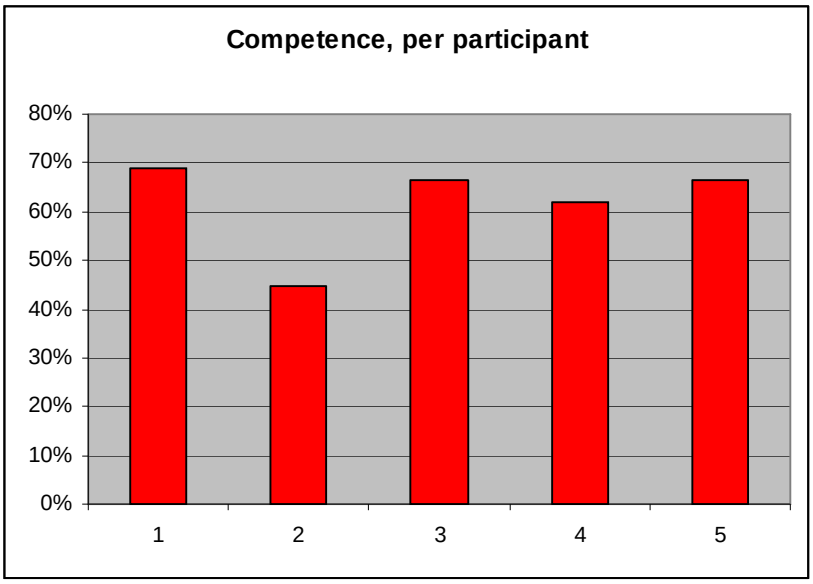


Figure 5-3: Survey results, competence, per participant

5.1.1.3 Discussion

Overall, the results of the competence part of the survey are at least pointing towards an agile way of working, with very few non agile results.
The total average of 62% can partially be explained by the fact that one of the respondents only scored 45%. The four remaining respondents actually averaged 66%, which by my definition is good in agile terms.

All of the participating companies seem to appreciate the importance of well educated staff, and it was pleasing to see that three of them actually do set aside at least two weeks a year for continuous education of their staff.

One of the participants do not seem to have their technical decisions being made at developer level, where that competence naturally should be, as much as they should to be considered agile. That same company does neither seem to have a management that is very open to suggestions from their developers. This is also considered non agile according to Schuh's table in my literature study.

5.1.2 Quality

5.1.2.1 Empirical data and analysis

Question number 10 intends to examine whether the focus is put more on quality rather than on delivery time, if the choice has to be made. The answers were distributed as; two on the second alternative, *"More focus on quality…"*, two on the fourth alternative, *"More focus on delivery time…"*, and one on the middle alternative, *"Equal focus on quality and delivery time"*. These results give me no clear indications in either direction. Though, a note should be taken on that none of the respondents chose any of the two extreme alternatives.

Question number 13, *"We think more testing only adds extra cost…"* had the answers widely distributed, though with an average result of 70% which should be interpreted as a high belief in test driven development. Further, also regarding the companies' faith in TDD, the answers to question number 12 give an average result of 70% as well. The results of both these questions point to a rather high degree of confidence in TDD.
The answers to question number 11, concerning how well known the concept of TDD is to the companies, results in an average of 70%.
Four out the five companies apply TDD in at least some projects, whereas one of the participants don't work test driven at all, as stated in the results of question 14. The answers to question number 15, regarding whether TDD is considered having great potential and worth investing in, resulted in the very high average of 85%. Even the company that stated that they don't use TDD chose the highest alternative on this question.

Question number 16 investigates whether the companies are willing to take a risk by trying out a new way of working. Four companies chose the next highest answer on this question; whereas the same company that don't apply TDD chose the next lowest answer here, resulting in an average of 65%.

Question number 18 states *"When we invest in something, we immediately want to be able to see the profit"*. Four of the participants chose the alternatives that correspond to 25% or 50%, whereas the non TDD company answered *"Strongly disagree"*, which is to be interpreted as the 100% agile answer.

Question number 33 was possibly somewhat tricky to interpret, or at least so was the purpose of the question. It states *"We put a lot of focus on finishing requirements and design before starting the implementation phase"*. This might sound like a positive thing, but on the other hand, this behavior fits well with non-agile methodologies like the waterfall model, and is therefore in this analysis considered as non agile. The results were widely distributed all the way from lowest to highest, with an average of 45%. I see no clearly speaking results here other than a note on the fact that some companies actually chose the very non agile alternatives.

On question 39, regarding the respondents' use of automated unit tests for code verification, three of the respondents answered that they often use unit tests, and the other two answered that they rarely use unit tests. Consequently, none of the companies chose the alternative "*never*", meaning that they are all in some extent at least familiar with and have tried the concept of automated unit testing. The average result on this question was 53%.

The average score on the quality questions in total was 60%.

5.1.2.2 Graphical presentation of the results

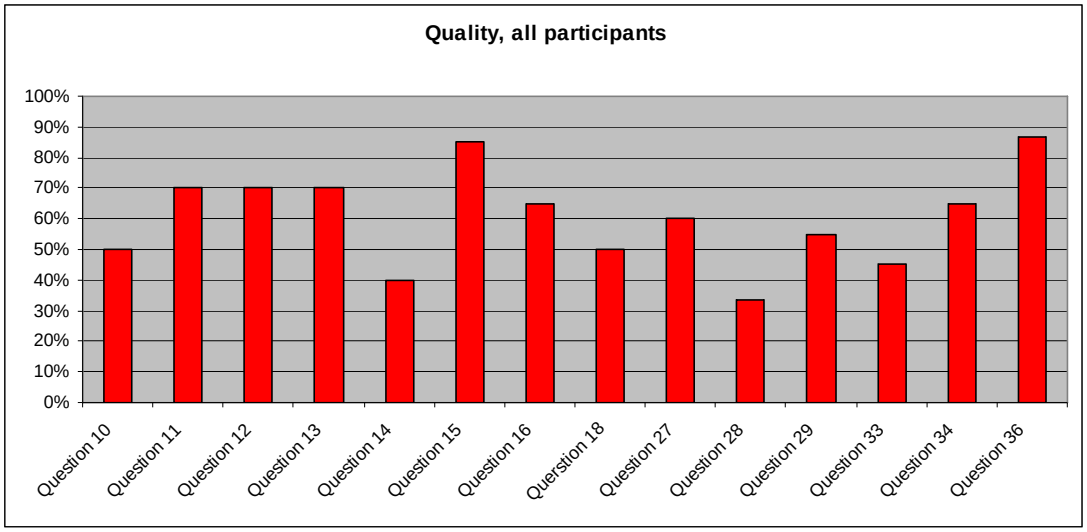


Figure 5-4: Survey results, quality, all participants

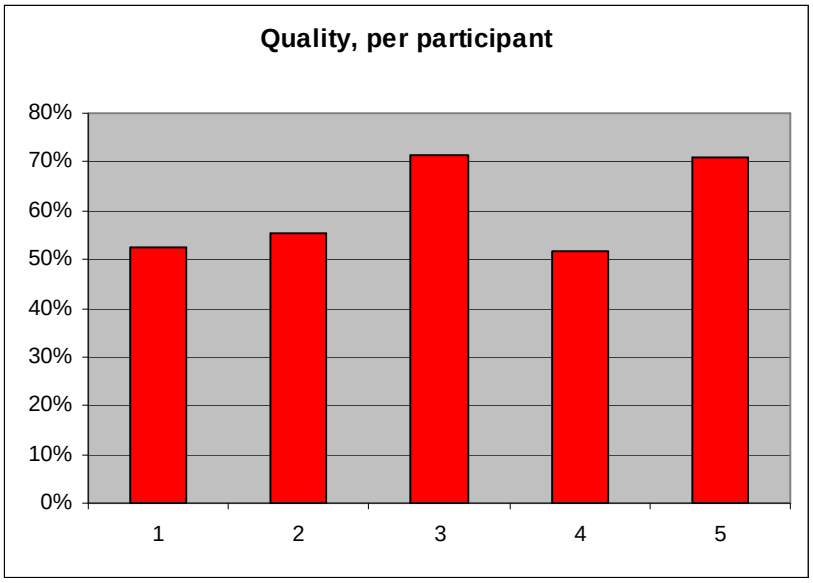


Figure 5-5: Survey results, quality, per participant

5.1.2.3 Discussion

Based on the results of question 10, I believe that this area leaves a lot of room for improvement. Understanding that a product of higher quality delivered a little later

than expected is to prefer before a flawed product delivered on time is an important step on the way of becoming agile. The company that chose the middle alternative actually wrote an explanation which suits well with Scrum philosophies; *"If delivery time is an absolute demand, we work on decreasing the scope instead of decreasing quality"*.

Questions 11 through 15 shows that the participating companies are well familiar with TDD, and they see the potential of working test driven. Even the one participant that don't work test driven at all have high faith in the concept. This company is the same company that showed low results on the management related questions in the competence section, so I can establish that as a probable reason; the developers see the potential, but are obstructed by a management unwilling to invest in a new way of working.

Question number 18 gave an interesting and unexpected result, as only one company chose an alternative saying that they do not agree with the statement, which is to be interpreted as seeing investments in a more long term fashion, which is part of the definition of being agile in my literature study.

Having several of the respondents choosing the alternatives stating that they do put a lot of focus on analysis and design, I definitely see a room for improvement here, as this behavior goes well with the waterfall model, which in turn is to be considered very non agile.

Based on the results on the question of usage of automatic unit testing it seems like all the participants have understood the benefits of the concept, but have not yet fully implemented it, so this area could also be improved.

5.1.3 Flexibility

5.1.3.1 Empirical data and analysis

An important criteria for being flexible, is having resources i.e. the developers, not too tightly coupled with specific systems, but rather having them move around between different systems. Question number 6 intended to investigate how this is thought of among the respondents. The result was an average of 55%.

Question number 7, regarding the ability to efficiently adjust and react when competitors change their behavior, resulted in the rather high average, 80%, which at least means that the respondents think of themselves as very flexible.

On question number 8, the results are distributed between alternatives all alternatives but 0%, with an average of 70%, meaning that the developers do not work separated from customers and product owners. This is of course positive in agile terms, since being agile means being able to quickly respond to your customers needs, and that is obviously easier to achieve with good communications between developers and customers.

Question number 9 aims to investigate whether the company has too much already invested in current methods to be willing to investigate in a new way of working. Most of the respondents chose an agile alternative on this question. The average result was 80%.

A question related to this is question number 17, which investigates the importance of the initial investment cost when it comes to decide on new ways of working. The result here was an average of 65%.
Question number 16 states *"We believe it is worth the risk of trying a new way of working"*. Four of the participants chose the fourth alternative, *"I mostly agree"*, whereas the fifth respondent did not agree with the statement. The average consequently was 65%.

The result of question 31, which states *"We put a great effort on following set deadlines"* resulted in four respondents agreeing to the statement, and the fifth company choosing the middle alternative, neither agreeing nor disagreeing. The average ended up very low, only 20%.

Question number 33 not only applies to quality but also to flexibility, and the result was as stated earlier 45%. Putting too much focus on design and requirements before starting the implementation really limits your flexibility.
Question number 34 aims to investigate whether the first releases of newly developed products tend to match the customer's expectations on the product. The result was an average of 65%.

The result of question 36 was that three of the respondents state that their customers are always represented in all their projects, and the other two respondents say that the customer is represented in many of their projects. The average result of that is 87%. Consequently none of the respondents chose the alternatives few or none.

The average score on the quality questions in total was 62%.

5.1.3.2 Graphical presentation of the results

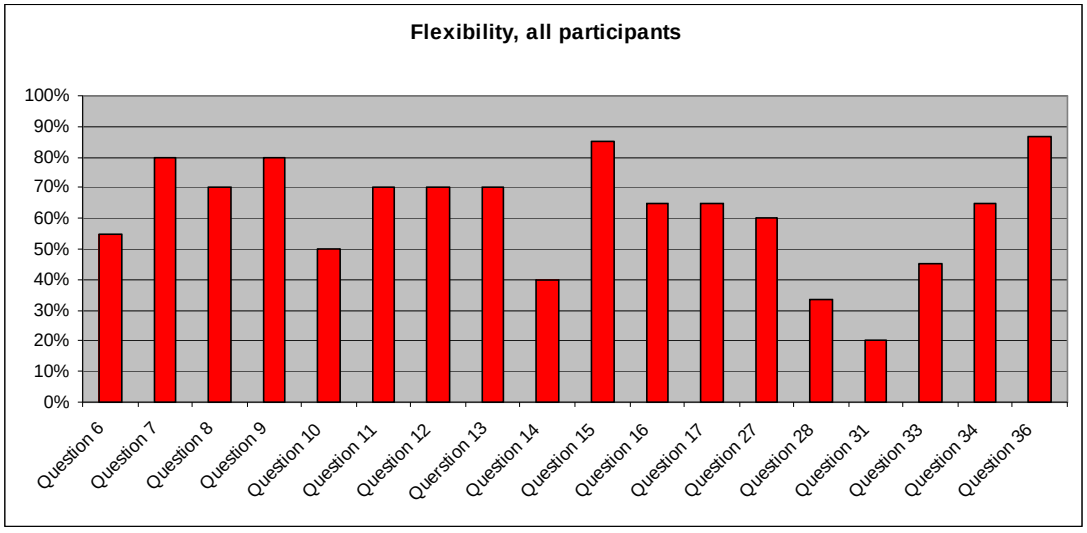


Figure 5-6: Survey results, flexibility, all participants

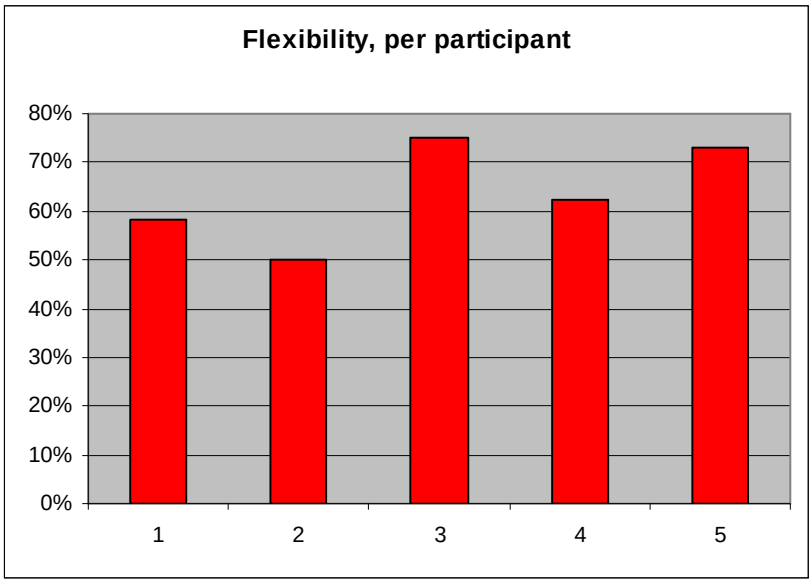


Figure 5-7: Survey results, flexibility, per participant

5.1.3.3 Discussion

Overall, the participants seem to well fulfill the criteria for being flexible according to my analysis model, for instance by putting value in not locking up developers too much in specific systems as stated by the results of question 6 and by maintaining good communications between developers and customers as shown by question 8.

Question 7 showed that all participants see themselves as very flexible, and that raises the following question. If you already think you are great at something, what will then motivate you to get even better?

Question 9 shows that the participants do not have any problems with investing in new ways of working due to previous investments, and question 17 indicates that they possess the ability to think in longer terms by not putting too much focus on the initial costs of investing in a new methodology. One respondent, however, states in question number 16 that they don't consider it being worth the risk of trying out a new way of working. This happens to be the same company that does not apply TDD at all.

The results of question 31 should be interpreted very negatively in this context, because it might indicate loss of quality to the benefit of faster delivery.

The high result on question 34 might be considered as being flexible since during the development, the customer's demands and requirements are likely to be changed, and thus being able to match those changes is by definition being flexible. This is closely connected to the good results on question 36, since having the customer represented on location during the development and thereby being able to quickly respond to changes in the customers requirements and any eventual new requests is also considered being flexible.

5.1.4 Communication

5.1.4.1 Empirical data and analysis

Question number 6, with the result of 55%, has already been brought up, both under competence and under flexibility, but I will mention it here as well, since it also regards communication.
Question number 8, with the result of 70% and already mentioned as used to investigate flexibility, by all means also investigates the use of proper communication; hence it is also mentioned here.

In question number 24, the purpose was to investigate any eventual differences between companies developing software for their own purpose or for other companies, i.e. external customers. Four of the respondents do mainly develop software for external customers while the fifth one mainly develops software for themselves. I have however not been able to link any differences of that company's results to this deviation.

To all of the five respondents, scrum is a well known concept, resulting in an average of 90% on question number 25, which is actually the highest value on all of the questions asked. Four of the companies apply scrum in some extent according to question number 26. One of the companies even chose the alternative stating that they use scrum in every single project. The average result was 53%.

Question number 27, *"XP is a well known concept to us"* resulted in an average of 60%, and question number 28 stated that four of the five respondents apply XP in some projects, however mostly still for trial purposes.

Question number 29 intends to investigate whether the company actively aims to internally communicate results and experience from ongoing project, thus distributing knowledge within the company. The result was an average of 55%.
The results of question number 32, which states *"We believe that the contact between our company and our customers should be kept higher than at a developer level"*, was an average of 40%, with no actually none of the participants choosing any of the two alternatives considered more agile.

Question number 37, regarding channels of communication between developers of a product and its' end users, resulted in an average of 63% among the four that actually answered the question, whereas the fifth respondent chose none of the fixed alternatives but instead wrote that *"that depends on the customer"*.
Question number 38 aims to investigate the communication between the developers and their management within their own company. The result was an average of 70%.

Question number 39, with the result of 53%, has already been mentioned in the quality section, but automatic unit tests should also to be thought of as an important means of communication, hence it is also brought up here.
The use of burndown charts was measured in question number 40, which showed that two of the respondents never use burndown charts, one of them use burndown charts seldom, and two of them use burndown charts often. The average result was 33%.

The average score on the communication questions in total was 59%.

5.1.4.2 Graphical presentation of the results

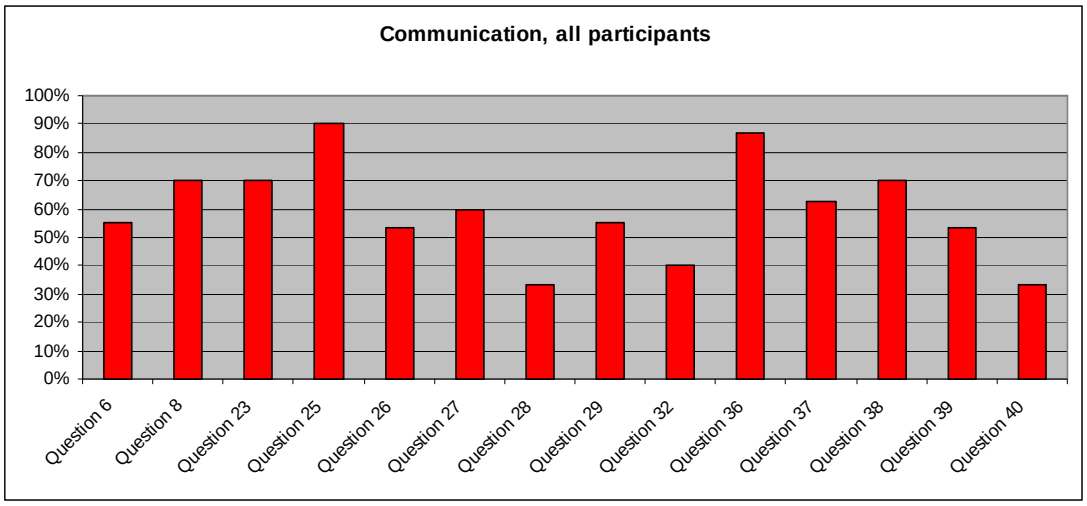


Figure 5-8: Survey results, communication, all participants

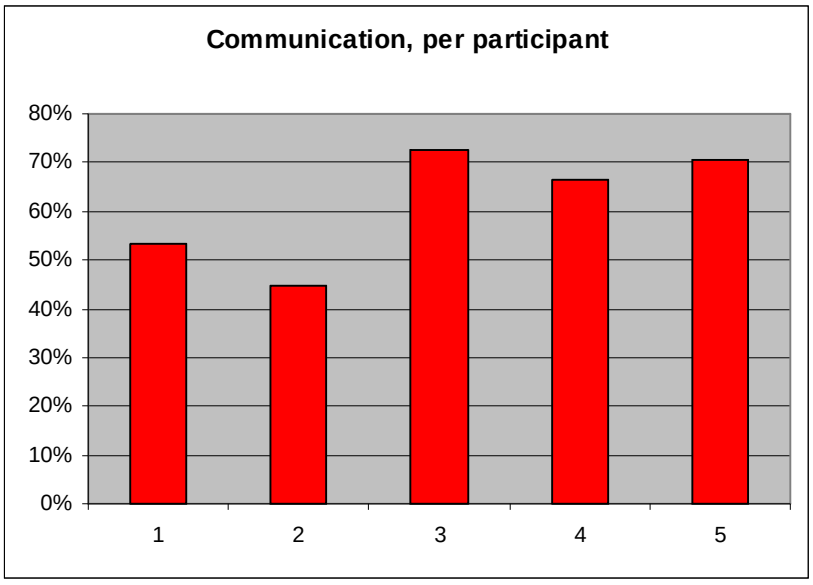


Figure 5-9: Survey results, communication, per participant

5.1.4.3 Discussion

As in the other three areas the participants show themselves to be on the right track also in the communication area, for instance by a very high result on the question of knowledge about Scrum, and also by the fact that four out of five apply scrum to some extent.
The knowledge of XP is not as good as the knowledge of Scrum, but still existing, and four companies do apply XP in some way.

One area that really should be improved in is internal communication, with only two out of five companies actually working on this according to the results of question 29.

A remarkable result is the one of question 32, which actually shows that none of the respondents think that the developers should be in contact with the company's customers. I realize however that this question might be unclearly formulated, and easily could be interpreted as to mean on more of a business level, which perhaps would justify the results better. Question number 37 on the other hand, is on the same topic, but probably better formulated, and the results of this question still shows room for improvement in the communication area.

The results of question 40 can be connected to the results of question 26, in the way that some of the participants have tried Scrum, but not to its full extent where burndown charts are essential, hence with an increased usage of Scrum, the use of burndown charts should also increase.

5.1.5 Cultural aspects

As stated earlier, some of the questions of the questionnaire are used for gathering background information, cultural aspects, on the investigated companies. This was done in order to try to identify differences in being agile between companies of different types regarding such cultural aspects.
The four cultural aspects taken into consideration are, as stated in chapter 3.3:

- Size
- Geography
- Degree of education
- Age

As shown in figure 5.1, those questions are questions 1, 2, 3, 4, 5, 20, 21, 22, 24 and 30. In the following section, I will shortly go through and analyze the answers provided to these questions.

To give an overview, the following table lists all the answers to these questions as well as the total average score of each participant.

	Participant	**1**	**2**	**3**	**4**	**5**
	Question					
Size	20	4	4	4	3	3
	21	3	3	3	3	3
	22	2	4	2	-	2
	30	1	2	1	2	2
Geography	1	3	3	1	1	3
	4	1	3	1	1	1
	24	2	1	1	-	1
Age	2	2	3	2	3	3
Education	3	2	3	2	2	3
	5	1	3	3	2	2
	Average agility score	55%	51%	70%	57%	70%

Figure 5-10:Answers to background/cultural aspect questions in relation to total agility score

5.1.5.1 Size

Questions 20 and 21 shows that all the participants have a high number of employees, and also that they all have more than 50 developers. Further, question number 22 resulted in a wide distribution of the share of consultants among the developers.
Question number 30 shows that the size of the developer teams within all of the participating companies usually never exceeds 10.
None of these results can however in combination with the respective total agility scores be interpreted as any indication of a relationship between size and degree of agility.

5.1.5.2 Geography

The answers on question number 1 shows that two of the respondents have mainly regional customers, while the other three have mainly international customers. A comparison of these two groups shows no obvious difference in terms of agility - of the two companies with regional customers, one scored relatively high, 70%, in total, and the other one relatively low, 57%, and in the group of companies with international customers, the distribution was similar, 51%, 55% and 70%.

On question number 4, four of the respondents stated that their IT business is run mainly from Stockholm, while the fifth stated that their IT is run mainly abroad. Indeed this company was the one scoring lowest in total, but not that much lower than some of the other participants that it would be reasonable to see this as a probable relationship.

Question number 24 resulted in a mixed distribution between all three alternatives, and no connection can be found here between being agile and developing software for neither your own company nor for external customers.

5.1.5.3 Age

Question number 2 investigates the average age of each company's employees, and the results are divided between two of the middle alternatives, 25-35 years and 35-45 years respectively. Just as with question number one, neither can any significant difference be shown between the two groups, as representatives of the groups scored both relatively high as well as relatively low in both cases.

5.1.5.4 Education

According to question number 3, all respondents have a share of employees with a university degree that is over 50%, and this can in combination with the total scores of the respondents not be seen as any clear indication in either way.

Question number 5 shows a wide distribution of how much time that is invested in further educating the staff within the different companies, and no real conclusions can be drawn here.

5.1.5.5 Discussion

As seen in the previous sections, no conclusions could actually be made regarding cultural aspects based on the data collected through the questionnaire.
This was however a rather expected result, as with as few as five respondents, the results would have to be very clear in order for me to be able to suggest any relations and indications. This was though not the case for this survey.

The question was raised whether I should include this section at all, since it actually produces no contributions in the means of results, but as I still see these factors as important to investigate, I have chosen to keep them in my analysis model and will propose a further investigation on the subject in a future, deeper study.

5.2 Case study

While the results of the questionnaire should be seen as an overview, an indication of what to expect of the general population, I will now go ahead and study a single company, with the intent of doing a deeper analysis of how that company has applied Scrum, XP and TDD in their organization.
I will try to identify things that went wrong, but also point out areas where the right approach has been taken.
I will also include a section of other important findings regarding agility, characteristics from my analysis model that can not be fit in to only one specific methodology out of the three.

The studied company is chosen out of the five questionnaire respondents, and thus is a company with over 100 employees, located in the Stockholm County.
The study will be done through observation and interviews.
Most of the information presented in this chapter is collected by interviewing my main contact within the company, the same person that filled out the questionnaire. The contact person is a senior software developer and architect and is well initiated in where the company stands regarding processes and methodologies.
Out of anonymity reasons, I will from now on mention my contact as “the architect”, and the studied company as “the company”.

As with the questionnaire, in order to bring more meaning to the observations, I will analyze the results while presenting them.

5.2.1 Scrum

In order to flexibly respond to changes, a solid platform for communication needs to be in place. This does according to my findings not yet exist within the company, at least not to a satisfying level.
An interesting example of that is that a serious attempt at introducing Scrum in the company was made in a single project approximately a year ago. During that trial period, which lasted about half a year, almost everything was in place, except for one very important piece of the puzzle, namely the product owner. Since the whole initiative of introducing Scrum was made at an organizationally speaking too low level, the product owners were not aboard, and since they consequently were not involved in prioritizing the work to be done, the whole concept failed according to developers that were part of the attempt.
I find this behavior repeated in many of the ongoing projects within the company also today, no matter the choice of methodology. The product owners are missing in the sense that they are never or rarely present physically. Most often their wishes and their feedback is communicated through project leaders and system administrators. This should be seen as a broken link in the communication within the company.

Today, no real Scrum goes on within the company. There are plans to start up a new initiative, but that will unfortunately be out of scope for this thesis because of time limitations.

Some of the important features of Scrum have been taken into practice though, as is the case with daily morning meetings.
In many projects, this has been and still is successfully used as a tool for communication between the team members.
The core of those meetings is internally referred to as an around-the-table session, in which all developers take turn and state what they did the previous day, what they intend to do during the rest of the current day, and also lift any eventual problems to the whole group. In some cases there is an outspoken development leader who is responsible for making sure issues with for example external systems are followed up on and taken care of.
By having these meetings, all team members will always know where the team is standing, and there naturally becomes a general openness within the team, where everyone is encouraged to collaborate with each other and ask for help when needed.

5.2.2 XP

XP is the one out of the three methodologies studied that has been used the least within the company. A short repetition of some of the important rules defined by XP says that:

- The customer is always available
- All code is pair programmed
- All code must have unit tests.

The first rule, customer availability, is, as stated earlier, definitely not fulfilled. In most of the ongoing projects, all contact the developers have with the actual customer is made via project leaders. The customer is never or rarely physically present during the development as stated in the previous section.

Pair programming is practiced now and then in different projects, but it is never really thought of as part of an actual methodology. Rather, it is done occasionally for instance when a developer faces a more challenging task, or a task where the solution is unclear and he or she sees the need of collaborating with a colleague in order to come up with the best solution together.

Unfortunately, a common view on pair programming within the company seems to be that it simply means that two developers perform a specific task in the same time as one developer could to that same task, and therefore is a waste of resources.

Unit testing the code is done to some extent in most projects, but never with full test coverage. I have observed several examples where the level of testing is very high in the beginning of a new project, but then decrease over time.
A possible explanation to this can be that as time often gets more critical during the later phases of development, testing has shown to be naturally down prioritized to the benefit of delivering the actual production code on time. This is indeed understandable under the circumstances of having fixed and very strict deadlines, but in no way acceptable in the means of being agile and quality-oriented.

Another key concept of XP, mentioned in my analysis model, simplicity, should also be considered. According to my observations this is rarely applied among the developers of the company today. Instead, a great effort is often put into trying to deliver very general and so called future proof solutions, solutions that do much more than what is actually required out of them. This might be considered a productive behavior, but it is actually in conflict with the definitions of both XP and TDD.

5.2.3 TDD and unit tests in general

A tool that if used correctly can help improve the quality of produced code is automated unit tests.
This is used to some extent within the company, but still leaves a lot of room for improvement before it can be considered a fully working practice.
For instance, in one of the larger and more important ongoing projects, a build server is actually in place, which constantly checks for updates in the repository containing all source code. If any changes are found, the build server automatically checks out the code, runs all the unit tests available, and produces a report of the results. This is indeed good, but unfortunately not enough.
Setting up a build server to perform the above described activities is a fairly easy one-time task. The challenge instead is to be consequent in writing unit tests for every new piece of code. This does not work very well, where some developers actually work hard to achieve complete test coverage and some do not take it that seriously at all. This is definitely something to become better at for the developers of the company.
In fact, if unit tests are not written for every single piece of functionality of the code, a build server report saying that everything works is actually very misleading, since the foundation of that statement is incomplete.
You might still have some use of the build server though, in the way that it will report on new code breaking tests that previously were successful.

Strictly test driven development however, built on the red-green-refactor rules described in my literature study, is not applied at all within the company. The will to develop applications test driven and good knowledge of TDD is found among a few dedicated developers, but in general the concept is misunderstood, as the common belief seems to be that TDD simply means “writing unit tests for your code”, both among developers and managers.

A possible issue could be the following:
A lot of the software development done within the company concerns modifying existing applications, and in some cases dealing with real legacy code. To suddenly go ahead and apply TDD on such a system is very hard, if not impossible.
The first and most obvious obstacle would lie in building up complete test suits for all existing code. This is a time consuming work and without having TDD anchored as a supported concept within the organization, funding for such work would be hard to get.
Based on this, I suggest that in order for the company to become successful in applying TDD, they would do best by starting off in a new project that does not rest on an existing code base. This should be done very strictly, with full test coverage, and full test-first approach, as defined by the red-green-refactor rules.
First after becoming successful in such a project, gaining valuable experience, the company should consider an attempt to apply TDD in existing systems.

5.2.4 General agile characteristics

Some of the characteristics observed can be found in the frame of reference, but do not necessarily belong to a specific methodology. These characteristics will be presented here.

5.2.4.1 Competence

According to the questionnaire, the share of employees within the company that has a university degree is above 75%, and according to the architect there is a company policy for recruiting people that more or less requires a degree out of the applicant. This tells me that the company really puts a high value in well educated staff.
The developers of the company are also experienced, as seen from the questionnaire, with an average working life experience of more than five years.
Most of the developers at the company have also been working within the company for a longer time and seem to enjoy their employer in general.

There are a lot of different projects running in parallel within the organization, most of those resting on a solid, already existing technical foundation, but as technologies change so do the software solutions.
If there is a need for developing a new solution in-house, the company puts a lot of faith in the skills and existing knowledge of their developers as all discussions leading to a new solution, how it should look, and which technologies it should consist of, always involve the concerned developers. This is a good and necessary property in the means of being agile.

In order to keep up with new technologies the developers are sent to various courses, approximately for a total of two weeks each every year. The topics of these courses range from programming languages to methodologies such as Scrum and TDD. There is however no regularity in this area, no schedules or such that are followed. It is rather up to each developer individually, or maybe a smaller team of developers, to communicate their needs of further education to their managers and get an approval to attend specific courses. This requires that all developers possess the ability to both identify areas of possible improvement, and also the will to admit this lack of competence and to be willing to improve and learn something new, which is not always the case.

5.2.4.2 Quality

In software development, the term quality rarely goes hand in hand with fast delivery, or put in another way, in order to implement a product of high quality, you would also need more development time than if the requirements on quality were lower. Suppose the deadline for delivery of a product is set and as that deadline comes closer the developers realize that they will not be able to deliver a product with as high quality as originally expected. Those developers then face two options, either they simply deliver what they can, and thus deliver on time, or they ask for more time to be able to deliver a better product. Ultimately the decision is lies of course not on the developers themselves, but rather on the stakeholders, the customers or the product owners, that is.
My perception of how this generally is handled at the company is that the deadlines are mostly valued higher than the actual quality of the product, which also matches

the architects view on the issue, as stated through his answer on question number 10 in the questionnaire.
Instead of postponing a release until the product really is finished, the way to do it more seems to make sure that a product with known flaws, but still considered good-enough for taking into use in a production environment, is released. These flaws are then taken care of in later releases.
A possible explanation to this behavior, and closely connected to question number 33 in the questionnaire, could be that within the company, all software development is done in a sequential manner, more or less according to the waterfall model. Emphasis is really put on finishing requirement specification and design documents before starting to implement the product, and once those documents are done, little or no flexibility is shown to any eventual desired changes.

5.2.4.3 Flexibility

To be flexible, from a developer's perspective, means essentially being able to respond quickly to changes in the previously stated requirements. In the company I have found both factors that help the developers on this point, and also factors that work against being flexible.
For instance, when it comes to increasing the distribution of knowledge of specific systems and general competence among the developers, there are some ambitions in some parts of the organization while other systems definitely have their dedicated developers.
Within the systems or groups of systems where the developers actually do work on improving each others competence, this is mainly an initiative from the developers themselves, rather than from their management.
The developers that only work on their own dedicated system are of course experts on those systems but know nothing about the surrounding systems, hence are not very flexible within the company. Also, if one of those developers would quit their job on short notice, the person that is to replace that developer would have to be taught from scratch which might take a long time. Such a situation could be avoided by having the developers rotated among several systems, thus distributing and sharing their competence continuously.

5.2.4.4 Communication

An example of communication is the link between the developers and the actual end user of the products developed. As the software developed in this company generally is used within the same company, this line of communication should be easier to maintain than if it should be towards external customers, but from what I have observed, that contact is still at a minimum.
According to my observations, the norm seems to be that product owners i.e. the customers rarely communicate directly with the developers of their system, but instead go through system managers and project leaders.
This behavior is dangerous in the way that it opens up for misunderstandings and confusion, obstacles that could easily be avoided by having the product owners working closer to their developers.

6 END DISCUSSION AND CONCLUSIONS

In this chapter, concluding the study, I will discuss my findings, pointing out key discoveries and also discuss the methods used.

6.1 General interpretation of the results

The overall results of the questionnaire indicate that the participants realize the value of becoming more agile, and they do possess many of the tools required but do not yet use them fully. In all four categories the total average scores were slightly below what I had defined as good in agile terms.
All participants are on the right track regarding education, both initial and continuous, but technical decisions still lie on a level above the developers with, by agile definitions, too much interference from respective management.
On average, the customer is represented throughout the ongoing projects in a very high degree, but looking at the individually studied company, this needs to be approved upon.
All participating companies in the study see themselves as very flexible, which I actually interpret as a dangerous characteristic, since by not being able to criticize your self you will not see in what areas you can get better, and you will thereby neither be motivated to improve.

6.1.1 Recommendations

The most important pointers on where to improve in general are:

- In almost all of the cases the three investigated methodologies, TDD, Scrum and XP, are applied only on trial and the participating companies do not seem to dare taking that step towards using agile methodologies exclusively. This can, and should be improved upon in order to become more agile.
- All but one of the participants put too high value in being able to immediately see return on their investments, instead of looking at investments in a longer perspective. This should be reconsidered for becoming more successful in the agile area.
- The focus on deadlines among the companies is very high. This property restricts a company's possibilities of becoming more agile and the participants need to review their look at this.
- The quality of developed code could be improved by increasing test coverage, preferably in combination with using a build server that offer test automation.

As for the company investigated closer in the case study, focus should lie on:

- Working out a good way to introduce TDD. This should initially be done in a new project with no dependencies on previous code and not in any of the existing projects with heavy legacy code bases.
- Opening up channels of communication between developers and product owners. A change of norm is really necessary here.
- Working out ways to redistribute existing knowledge among their developers, as a suggestion by reevaluating XP as an accepted way of working so that it can be applied to a higher degree than today.

- Introducing Scrum again, but this time doing it fully. It seems like this will become a reality shortly after this thesis is finished, and it would be very interesting to follow up on that process.

6.2 A critical view on the methods used

Overall, I have found a combination of a quantitative study and a qualitative study to be a working concept. I have however come to some conclusions and insight on where to improve in any further studies.

As the original intent of this study was to perform a larger quantitative study, a couple of cultural aspects were brought in to the analysis model. This was done with the intent of trying to identify possible reasons for any eventual differences between the respondents, such as for instance investigating whether the average age of the developers would have an impact on how agile a company would be.
However, as the study ultimately was performed on a relatively small number of participants, statistically speaking, no such conclusions could be made regarding cultural differences.
I still chose to keep these characteristics in my analysis model, and the questions are also kept in the questionnaire since I still believe that this aspect would be interesting to investigate in a future deeper study on the topic.

While the analysis of the results of the questionnaire went on fairly easy, I still see room for improvement in the way that all questions attempting to measure the same characteristic (in this case all non background questions ultimately measured agility) could for comparison reasons all have the same number of alternatives to choose from.

7 SUMMARY

Being agile in the field of software development can be defined as fulfilling a number of criteria in the four areas communication, quality, flexibility and competence.

By performing this study, investigating those criteria, I have shown that the companies involved are all on the right track, as they show themselves to be willing to work agile. However, possessing the will is not sufficient - they all have a lot of work remaining, as none of them could actually be considered very agile according to my survey.

Communication should be improved upon, where product owners working in closer collaboration with the developers should be a prioritized area.
Another weakness has been identified in the lack of means to distribute existing knowledge among internal resources.

The developers of all participating companies do test their code, but in none of the cases with full test coverage. In order to achieve higher quality, this should be focused upon as well as considering taking automated build servers into use.
The companies need to see testing as a natural and necessary part of developing new code instead of seeing it as a verification tool used at the end of the development phase.

While a couple of the participants seem to put quality before fixed delivery times, the others prioritize deadlines higher and all participants actually stated that they put a lot of effort on following deadlines overall. This should be considered an area of improvement – perhaps the companies should be more flexible in their time estimates, leaving a wider margin of error.

All of the investigated companies do have experienced staff, with a high rate of employees with a university degree.
Also the level of continuous education within the companies is for most of the participants sufficient, but could and should be increased in some cases.

Another very important lesson learned from the case study is that introducing agile methodologies in a company should be done from a management perspective, to make sure that all concerned parts are properly involved.

8 FUTURE RESEARCH

As the questionnaire used in this study was only sent out to five companies, that survey can only be considered to be an indication of what to expect from the population as a whole.
No real mathematical and statistical analysis can nor should be done on such a small selection, hence the next natural step from the research done in this thesis would be to perform the study in full scale, attempting to establish statistically reliable and valid results.

Initially, such a study should still be concentrated to one city, preferably Stockholm, since that is where the largest base of companies in the software development business of Sweden is found, but in a longer term it would also be interesting to perform the study nationally, and try to find geographical variations.

Another interesting area of research would be to conduct another case study, investigating the whole progress of a company while introducing an agile method, from the very beginning.
This would mean first finding a company that has never before worked with a specific methodology, for instance scrum, and then follow that company's progress while introducing scrum, measuring eventual improvements and identifying issues and problems as they appear.

As both the qualitative and the quantitative study in this thesis was presented more or less from a developers perspective, it would be interesting to do a comparison between these results and how these issues are perceived at management level.

REFERENCES

[1] Erdogmus, H., Morisio, M., Torchiano M, On the Effectiveness of the Test-First Approach to Programming, http://iit-iti.nrc-cnrc.gc.ca/publications/nrc-47445_e.html (Cited 2009-01-19)

[2] Ford, H (1922) My Life and Work
e-book available at http://www.gutenberg.org/dirs/etext05/hnfrd10.txt
(Cited 2009-01-19)

[3] The world's most valuable brands 2007
http://www.gizmag.com/go/7385/ (Cited 2009-01-19)

[4] The 7 manufacturing wastes
http://www.emsstrategies.com/dm090203article2.html (Cited 2009-01-19)

[5] The agile manifesto
http://www.agilemanifesto.org/ (Cited 2009-01-19)

[6] Lean software development
http://www.poppendieck.com/ (Cited 2009-01-19)

[7] Kniberg H, Scrum and XP from the Trenches, 2007, InfoQ

[8] Cockburn A, Agile software development 2. ed 2007, Addison Wesley

[9] Extreme programming – a gentle introduction
http://www.extremeprogramming.org/ (Cited 2009-01-19)

[10] Beck K, Test-driven development by example, 2003, Addison Wesley

[11] Patel R, Davidson B, Forskningsmetodikens grunder, 2003, Studentlitteratur

[12] Eriksson L T, Wiedersheim-Paul F, Att utreda forska och rapportera, 2001, Liber AB,

[13] Kylén J-A, Fråga rätt, 1994, Kylén Förlag AB

[14] Ziv H, Richardson D, 1997 "The Uncertainty Principle in Software Engineering" Proceedings of the 19th International Conference on Software Engineering.

[15] Boehm P, Papaccio P, 1998 "Understanding and Controlling Software Costs" IEEE Transactions on Software Engineering, Oct. 1988

[16] Larman C, 2004, "Agile and iterative development - A manager's guide", Addison Wesley

[17] Goldman S, Nagel R, Preiss K, 1997, "Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer", John Wiley & Sons

[18] Corporarate Report, 2003, "Agile methodologies survey results", Shine Technologies Pty Ltd., Victoria Australia

[19] Thomas M, 2001, "IT Projects Sink or Swim", British Computer Society Review

[20] Cohen D, Larson G, Ware B, 2001, “Improving Software Investments through Requirements Validation.” IEEE 26th Software Engineering Workshop

[21] Holme I M, Solvang B K, 1997, ”Forskningsmetodik – Om kvalitativa och kvantitativa metoder”, Studentlitteratur

[22] Johnson R, Hoeller J, 2004, “J2EE Development without EJB”, Wrox

[23] Schuh P, 2004, “Integrating Agile Development in the Real World”, Charles River Media

[24] SCB’s företagsregister 2008
http://www.scb.se/Grupp/Foretagsregistret/_Dokument/Broschyr2008.pdf
(Cited 2009-01-19)

[25] Research knowledge base
http://www.socialresearchmethods.net/kb/scallik.php (Cited 2009-01-19)

[26] Wikipedia
http://en.wikipedia.org/wiki/Waterfall_model (Cited 2009-01-19)

APPENDIXES

Appendix A – Survey Letter

[English translation of the letter, originally written in Swedish]

Stockholm October 15 2008

Hello!

My name is Daniel Barke and I am a student at Blekinge Institute of Technology in Karlskrona/Ronneby. I am currently writing my D-level thesis within computer science, which will conclude my masters degree in that area.
The thesis is based on the upcoming results of the attached questionnaire, which I really would appreciate if You could take the time to fill out.
I estimate the time it will take to fill the questionnaire out to approximately ten minutes.

The survey makes up the base of an investigation of preparation for and attitude towards agile ways of working within software development among larger companies in the Stockholm area, an investigation that I will be working on during the calendar year 2008, 20 hours a week. The thesis will be presented in January 2009.

No reference to any specific company will be made, and all answers will be treated anonymously and confidentially. Only aggregated results of all answers will be presented.

By filling out this survey You will increase the reliability in my study and also give valuable information to decision makers within companies developing software. As far as I know, no similar study has been made before.

Your company was one of the ones chosen randomly among larger companies within the Stockholm area. I off course guarantee full secrecy, since the purpose of this study is to investigate the aggregated market and no specific companies.

I would appreciate if You could return the filled out questionnaire to me using the pre-franked envelope supplied before October the 31st, 2008.

Best regards

Daniel Barke
Student at Blekinge Institute of Technology

Appendix B – The survey

[English translation of the questionnaire, originally written in Swedish]

Studying companies' preparation for and attitude towards agile ways of working

This questionnaire is a part of my D-level thesis in the subject of computer science, which I am working on at Blekinge Institute of Technology. In this survey I intend to investigate the preparation for and attitude towards agile ways of working within software development among larger companies in the Stockholm area. All information supplied by You will be treated confidentially and no reference to any specific company will be made.

I estimate the time it will take to answer these questions to approximately ten minutes.
If You find anything unclear regarding any of the questions, or regarding the questionnaire in general, don't hesitate to contact me, Daniel Barke, through either e-mail at daniel.barke@gmail.com or through phone. My phone number is xxxx-xxxxxx.
Please answer the questions in the way that You think matches Your company. If You should feel that any of the questions don't apply to Your company, please try to find the most appropriate alternative anyway.

Please return your answers to me in the supplied envelope at latest the 31st of October, 2008.

Thanks in advance!

Grading:

I strongly agree	5
I mostly extent	4
I neither agree nor disagree	3
I mostly disagree	2
I strongly disagree	1

Only one alternative per question should be marked.

Question 1:
Our customers are mainly:
☐ Regional
☐ Domestic
☐ International

Question 2:
Approximately, within which interval is the average age of the employees of your company?
☐ below 25 years
☐ 25 – 35 years
☐ 36 – 45 years
☐ 46 – 55 years
☐ above 55 years

Question 3:
Approximately, how many of your employees have an academic degree? (from university).
☐ 100 %
☐ 75-99 %
☐ 50-74 %
☐ 25-49 %
☐ 1-24 %
☐ 0 %

Question 4:
Our IT business is run mainly from:
☐ The Stockholm area
☐ Another part of Sweden
☐ Abroad

Question 5:
We dedicate time and resource for further education of our staff.
☐ More than two weeks a year
☐ Two weeks a year
☐ One week a year
☐ Less than one week a year
☐ Never

Question 6:
We work actively to improve the developers' possibilities to be able to share knowledge and learn from each other between different teams by having a flexibility of developers between different systems instead of specified systems for each developer.
☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 7:
We think that we can adapt quickly and efficiently when our competitors and customers change their behaviour.

☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 8:
Our developers work separated from customers and product owners.

☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 9:
We think that it is unprofitable to invest in new ways of working because we already have a lot invested in current methodologies.

☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 10:
If we must make a trade-off between quality and delivery time we put

☐ focus mainly on quality
☐ more focus on quality than in delivery time
☐ equal focus on quality and delivery time
☐ more focus on delivery time than on quality
☐ focus mainly on delivery time

Question 11:
Test driven development is a well known concept to us

☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 12:
We are uncertain of how to be able to benefit from test driven development.

☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 13:
We think that more testing during development only adds extra cost.

☐ ☐ ☐ ☐ ☐
1 2 3 4 5

Question 14:
Today we work with test driven development
☐ Exclusively
☐ In some projects
☐ On trial in a few projects
☐ We do not work with test driven development

Question 15:
Test driven development feels like something well worth investing in in the future.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 16:
We believe that it is worth the risk of trying a new way of working.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 17:
The direct investment cost is the heaviest factor when deciding whether to introduce a new way of working or not.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 18:
When we invest in something, we immediately want to be able to see the profit.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 19:
The average working life experience among developers within our company lies approximately within the following interval:

☐ below 2 years
☐ 2 – 5 years
☐ 5 – 10 years
☐ above 10 years

Question 20:
The total number of employees in our company lies within the following interval:

☐ 0 - 19
☐ 20– 99
☐ 100 – 200
☐ above 200

Question 21:
The number of developers in our company lies within the following interval:

☐ 0 - 9
☐ 10 – 50
☐ above 50

Question 22:
The share of employees (as opposed to consultants) of our developers lies within the following interval:
☐ 0-25%
☐ 25-50%
☐ 50-75%
☐ 75-100%

Question 23:
Our developers do have a lot to say regarding choice of methodologies and tools.
☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 24:
The end user of the software that we develop is mainly

☐ External customers
☐ Our own company
☐ Evenly distributed between the two alternatives

Question 25:
SCRUM is a well known concept to us
☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 26:
Today we work with SCRUM
☐ Exclusively
☐ In some projects
☐ On trial in a few projects
☐ We do not work with SCRUM

Question 27:
XP is a well known concept to us
☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 28:
Today we work with XP
☐ Exclusively
☐ In some projects
☐ On trial in a few projects
☐ We do not work with XP

Question 29:
We work actively within the company to communicate results and experience from ongoing projects.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 30:
The size of the developer teams within our projects normally lie within the following interval:

☐ 1-4
☐ 5-10
☐ Above 10

Question 31:
We put a great effort into meeting set deadlines

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 32:
We believe that the contact between our company and our customers should reside higher than on developer level.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 33:
We put a lot of focus on finishing requirements and design before starting the implementation phase.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 34:
Our newly developed products match the customers wishes already on the first delivery

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 35:
When introducing a new technical solution, the decision making resides mainly

☐ On management level, without influence of the concerned developers
☐ On management level, with some influence of the concerned developers
☐ On developer level, with final approval from the management
☐ Exclusively on developer level

Question 36:
The customer is represented in some way all the way from the beginning until the end

☐ In all of our projects
☐ In many of our projects
☐ In a few of our projects
☐ In none of our projects

Question 37:
We have good channels for communication between developers and end users of our products.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

Question 38:
We believe that the management is perceptive to wishes and suggestions of improvement from our developers.

☐ ☐ ☐ ☐ ☐

1 2 3 4 5

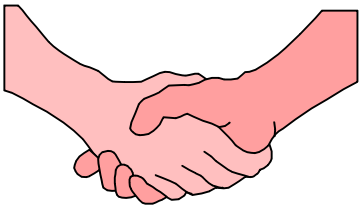
Question 39:
Our developers use automated unit tests to verify code changes

☐ Always
☐ Often
☐ Rarely
☐ Never

Question 40:
We use burndown charts to present and communicate the success of our projects

☐ Always
☐ Often
☐ Rarely
☐ Never

Thanks for taking the time to fill out this questionnaire



Appendix C – Survey results

Raw results

The following table lists all answers gathered from the survey. The answers are simply numbered by the order in which they are listed in the actual questionnaire, so that the first listed alternative is always numbered 1, the second is number 2 and so on.

Participant	1	2	3	4	5	
Question						**Std dev.**
1	3	3	1	1	3	1,10
2	2	3	2	3	3	0,55
3	2	3	2	2	3	0,55
4	1	3	1	1	1	0,89
5	1	3	3	2	2	0,84
6	4	2	3	3	4	0,84
7	5	4	4	4	4	0,45
8	3	4	1	1	2	1,30
9	3	2	1	1	2	0,84
10	4	4	2	3	2	1,00
11	4	4	4	3	4	0,45
12	2	3	2	3	1	0,84
13	2	3	1	4	1	1,30
14	3	4	2	3	2	0,84
15	5	5	4	3	5	0,89
16	4	2	4	4	4	0,89
17	4	1	2	2	3	1,14
18	4	1	3	3	4	1,22
19	3	3	-	3	3	0,00
20	4	4	4	3	3	0,55
21	3	3	3	3	3	0,00
22	2	4	2	-	2	1,00
23	5	3	4	3	4	0,84
24	2	1	1	-	1	0,50
25	5	4	5	4	5	0,55
26	3	4	2	2	1	1,14
27	2	4	4	4	3	0,89
28	3	4	4	2	3	0,84
29	2	3	4	3	4	0,84
30	1	2	1	2	2	0,55
31	4	3	5	5	4	0,84
32	4	3	3	4	3	0,55
33	4	3	3	5	1	1,48
34	4	4	4	3	3	0,55
35	3	2	3	-	3	0,50
36	2	1	1	1	2	0,55
37	3	3	4	-	4	0,58
38	4	3	4	4	4	0,45
39	3	3	2	2	2	0,55
40	5	4	2	3	2	1,30

Translated results

The following table shows the results on all non-background questions of the questionnaire, translated into percentages. The reason for not showing the background questions in this table is that those questions are not graded and analyzed in percentages, hence they do not fit into this table.

Participant	1	2	3	4	5	
Question						**Average**
3	75%	50%	75%	75%	50%	65%
5	100%	50%	50%	75%	75%	70%
6	75%	25%	50%	50%	75%	55%
7	100%	75%	75%	75%	75%	80%
8	50%	25%	100%	100%	75%	70%
9	50%	75%	100%	100%	75%	80%
10	25%	25%	75%	50%	75%	50%
11	75%	75%	75%	50%	75%	70%
12	75%	50%	75%	50%	100%	70%
13	75%	50%	100%	25%	100%	70%
14	33%	0%	67%	33%	67%	40%
15	100%	100%	75%	50%	100%	85%
16	75%	25%	75%	75%	75%	65%
17	25%	100%	75%	75%	50%	65%
18	25%	100%	50%	50%	25%	50%
23	100%	50%	75%	50%	75%	70%
25	100%	75%	100%	75%	100%	90%
26	33%	0%	67%	67%	100%	53%
27	25%	75%	75%	75%	50%	60%
28	33%	0%	33%	67%	33%	33%
29	25%	50%	75%	50%	75%	55%
31	25%	50%	0%	0%	25%	20%
32	25%	50%	50%	25%	50%	40%
33	25%	50%	50%	0%	100%	45%
34	75%	75%	75%	50%	50%	65%
35	67%	33%	67%	-	67%	58%
36	67%	100%	100%	100%	67%	87%
37	50%	50%	75%	-	75%	63%
38	75%	50%	75%	75%	75%	70%
39	33%	33%	67%	67%	67%	53%
40	0%	0%	67%	33%	67%	33%
Average	55%	51%	70%	57%	70%	

Total average: 61%