Identifying the level of agile in software development projects and what will be the effect?

Niels Oomen, BEng

Student number: 360324

The Thesis Committee for Niels Oomen
Certifies that this is the approved version of the following thesis:

Identifying the level of agile in software development projects and what will be the effect?

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor:
dr.ir. Ton van der Wiele

Co-reader:
dr. Paul Beije

Identifying the level of agile in software development projects and what will be the effect?

by

Niels Oomen, BEng

Master Thesis

Presented to the Rotterdam School of Management for the Degree of:

Master of Science in Business Administration

Rotterdam School of Management

Erasmus University

August 2013

© www.dilbert.com

Preface

The master thesis, in front of you, is the end result of the part-time study Master of Science Business Administration at the Rotterdam School of Management, Erasmus University. With this thesis a period of two years will be finished. An intensive period with evening colleges two times a week and finding a balance between study, personal life and a fulltime job. It was a long and sometimes a difficult journey, but I'm glad I did it. It resulted in new knowledge (also about myself), but also in new friends.

It is not possible to fulfill this study without help, so from this position I would like to thank some people. First I would like to thank my colleagues of Transtrack International and especially Ronald van Vliet and Theo DeOliveira who gave me the opportunity to follow this education and give me the time to do this. Second I would like to thank my coach, Ton van der Wiele for his input and feedback. Especially during the moments I was lost or blind for possible problems. Also I would like to thank my co-reader Paul Beije for his feedback and comments.

Further I would like to thank my family and friends, who accepted my absences and lacks of being family or a friend the last two years or who helped me to succeed this study with all their help in their own way.

Last, but not least I would like to give special thanks to my wife and my sons. Riana, without your sacrifices the last two years I was not able to this study at all. Thank you for all your help, input and encouragements to me. Twan, from now on I will spend unlimited hours in the swimming pool with you. Roel, you only will hear the stories about daddy's absences for his study from your big brother, you will never experience it.

Thank you all so much.

Niels Oomen
August 12th, 2013

Executive summary

We live in a society that is and will be a more and more digital environment, both private and work related. Most of the products we use contain software or we use a software system. It is a discussion for several years now how the software development process should be controlled to deliver the software faster, better and cheaper. The most common way to deliver software is via the waterfall method. This way of working means decide early and deliver slowly. This approach of working has been changed into a more adaptive way, this means decide late and deliver fast. This last method is called: Agile. The purpose of this study is to investigate the effects if a higher level of agile will increase the quality of a software development projects.

For this study the chosen research strategy is a survey. The major objective of this research is to test the probabilistic relation between the level of agile software development projects and four elements of a software development project: quality of the software, project finished within budget, project finished within budget and project finished within scope. The data for this research were gathered by using an online survey with 33 questions. The empirical part of this research was conducted in between June and July 2013. The subjects were selected based on agile user groups at LinkedIn.

Based on the data analysis it can be concluded that there is no significant relationship between the project elements (budget, planning, scope and quality) and the level of agile. Looking to the initial research question: “Will the, level of the agile working method, influence the quality of a project?” the results did not support the expectations that a higher level of agile will increase the total quality of a project. Although the outcome is not significant some directions of the relations can be are interesting. Therefore it is the advice to do further research to the level of agile and the relation between the quality of software development projects in future studies.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

This chapter will give an introduction about this research and will contain five paragraphs. The first paragraph will provide a general introduction of the research. It follows with the research questions and the conceptual model. The third paragraph will discuss the theoretical and practical relevance of this research. Then the research methodology is described and the last paragraph explains the structure of this thesis.

1.1 INTRODUCTION

We live in a society that is and will be a more and more digital environment, both private and work related and most of the products we use contain software. As a result of this, all kinds of projects will have a relationship with IT or contains an actual software implementation. This can be a brand new software application or an upgrade of a current system. We experience this every day. It can be an upgrade of the Microsoft Office software at home, the warehouse inventory system at work or an update of an app on our smartphone or tablet. The enormous amount of products, which includes software, is too large to describe. Without knowing it, software is all around us: computers & smartphones, plains, trains, equipment in hospitals, ATM machines, etc. The list of products we use, where software is involved, is just extremely long. It's what lets us get cash from an ATM, make a phone call, and drive our cars. Just to give an example based on (Goldstein, 2005): General Motors Corp. estimated that by 2010 its cars would each have 100 million lines of code. The user is not thinking about the software of the car, the user is expecting a perfect car to drive from A to B.

We use software direct or indirect and work or private related on a daily basis. How come that, those IT implementations often result in a failure? This means that the

implementations were delayed, cancelled or abandoned (Charette, 2005) and (Goldstein, 2005). Based on research 32% of the IT projects are successful (on time, within budget and fully functional) (Standish Group International, 2010). This means that 68% of the IT projects are not successful! A claim is often that the software development process is not perfect (Chow & Cao, 2008). The manner in which the software development takes place is mostly time consuming and the different projects are delivered with a delay or cancelled.

It is a discussion for several years now how software development should be controlled to deliver the software faster, better and cheaper (Dybå & Dingsøyr, 2008). According to (Petersen & Wohlin, 2009) it is much more important to be more flexible in handling changes of requirements, so software can be delivered quickly to the market.

The most common way to deliver software is via a predictive way, which is called the waterfall method (Royce, 1970). This way of working means decide early and deliver slowly. This approach of working has been changed into a more adaptive way, this means decide late and deliver fast. This last method is called: *Agile*. Based on a survey in 2008 among 3061 companies in 80 countries, 25% of the respondents realized more than 25% improvement on productivity and 30% of the respondents realized a reducing of software defects by more than 25% (Versionone, 2008). Looking to the situation that agile software development is more adopted in the software development society over the last years and has been successful (Versionone, 2008) does it really make a difference and how will it make difference? Will agile software development methods increase the quality of a software development projects?

1.2 RESEARCH QUESTIONS AND CONCEPTUAL MODEL

The objective of this research is to contribute to the theory about the level of agile working methods and the aspects (dependent concepts) of a project. The starting point for

all propositions will be the same: *The level of the agile working method*. All this will lead to the following general research question:

Will the, level of the Agile working method, influence the quality of a project?

This study will look into agile software development projects and what the results are by using this method within a project. It is not a comparison between agile and non-agile projects, because it is hard to compare two different development methods within the same project. The research can be split up in two parts: 1) Agile development methods and 2) project management concepts: budget, scope, planning and quality. First the literature review will have a focus on agile software development with a brief overview of software development in general. Second the literature review will have an outline of the concepts of project management related to software projects, also known as the Iron Triangle (Atkinson, 1999). The literature review will lead towards the following propositions:

- *P1: When the level of the "agile" method is higher, then it is likely that the quality of the software higher*.
- *P2: When the level of the "agile method is higher, then it is likely that the project will finish more closely to the estimated budget*.
- *P3: When the level of the "agile" method is higher, then it is likely that the project will finish more closely to the estimated deadline (planning)*.
- *P4: When the level of the "agile" method is higher, then it is likely that the project will finish more accurate within in the defined scope*.

Based on the exploration of the literature about agile software development methods and the concepts of project management I have the claim that the higher the level of the used agile software development method, the better the result of each aspect of the project management will be. This all will lead towards the following conceptual model:

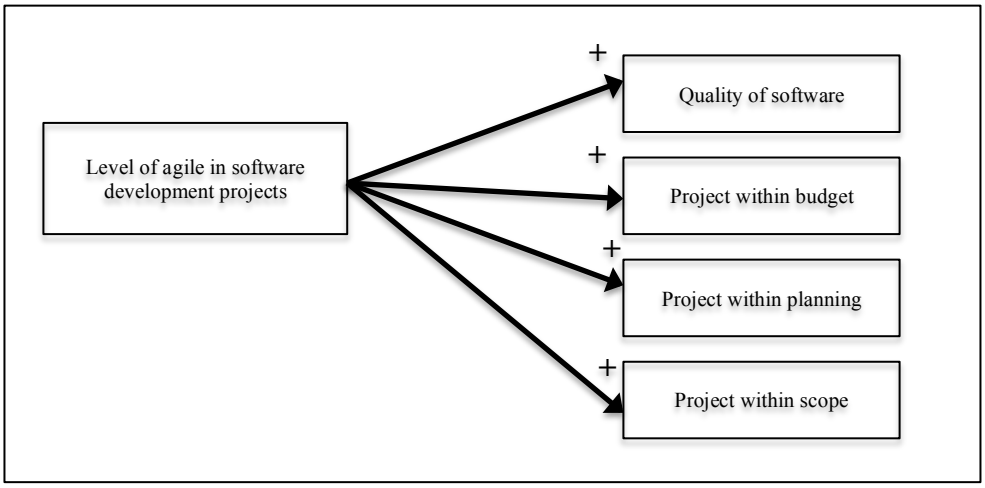

Figure 1: Conceptual model

1.3 THEORETICAL AND PRACTICAL RELEVANCE

This research will have a theoretical relevance and a practical relevance. The practical relevance will look to the results of implementing an agile working method for projects in general and how will this have an effect on the quality of a project and specific for agile software development projects. Further this thesis will support the theory related to agile software development projects and how quality of a project and the effect of the agile working method implementation can be measured and if a agile method will result in better quality, like mentioned in the literature.

1.4 RESEARCH METHODOLOGY

The methodology for this research will be a survey. This research will test the probabilistic relation between the level of agile working methods in software development projects (independent concept) and the four dependents concepts of the projects: 1) quality of the software, 2) project finished within budget, 3) project finished within budget and 4) project finished within scope. This research will be executed based on the research approach, which is defined by (Dul & Hak, 2008) and visible in Table 1: Research design.

An experiment is the preferred research strategy (Dul & Hak, 2008), but will not be feasible due to the available time. The current research needs to be done within six months. Developing software will take longer than six months. The second reason for not using the experiment as a research study is because of testing a probabilistic relation. To generalize the outcome a large number of instances ("Large N") are needed. The best way to do is by using a survey a research strategy. A survey strategy will be used during this research. To collect the data an online questionnaire tool will be used. The instances will be found via specific focus groups within the agile community. An example will be to use social media (LinkedIn) to contact project managers of software projects who are using agile methods. The limitation is that software projects need to be executed based on agile methods.

Phase	**Research project step**
Preparation	1) Define research topic
	2) Define the general research objectives and the general type of research
	3) Define the specific research objectives and the specific type of research
Research	4) Choose the research strategy
	5) Select instances
	6) Conduct measurement
	7) Conduct data analysis
Implications and report	8) Discuss results
	9) Report the research

Table 1: Research design
Source: (Dul & Hak, 2008)

1.5 STRUCTURE OF THE THESIS

This thesis is having five chapters and will have the following structure. Four chapters follow after this first introductory chapter. Chapter two will present the literature review, the final propositions and conceptual model. Chapter three describes the research methodology. The data analysis will be presented in chapter four. The final chapter will contain the conclusions and discussion.

Chapter 2: Literature review

The objective of the chapter is to present a review of the available literature relevant for this research. This chapter will begin by giving a historical explanation about software development in general. In the second paragraph there will be a clarification about agile software development. This clarification will contain a general introduction, advantages and characteristics of agile software development. In the third section software development quality will be discussed and in paragraph four the relation between software development and project management will be explained. The last part of this chapter will be a summary of the theory including the propositions.

2.1 SOFTWARE DEVELOPMENT

The terminology of software development or software engineering was initially defined at the NATO Science Conference in 1968 (Ehlers, 2011), but has been changed / updated over time. In this thesis the definition of software engineering or software development of (IEEE, 1990) will be used:

> 'The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.'

Software development is not only writing code, but includes also the whole software development process (Ehlers, 2011). Since the introduction of software development in the late 60's, several methodologies have been introduced (Huo, Verner, Zhu, & Babar, 2004). According to (Cugola & Ghezzi, 1998) the waterfall method is the most broadly known method in the software development community. The waterfall method is introduced by (Royce, 1970) and contains the following general steps in the software development process: Requirements, Design, Implementation, Validation and

Maintenance. The basic principle is that the waterfall development method proceeds in a linear approach (Cugola & Ghezzi, 1998), is that this process is not a cycle. According to (Brooks, 2012) and (Hoogendoorn, 2012) one of the main problems is that it is almost impossible to go a step back to the previous step. The phases are a one-off and need to be executed in the correct sequences. The waterfall method is a method that is sufficient for extending a known solution (Brooks, 2012). In Figure 2: Waterfall model the waterfall method is visualized.

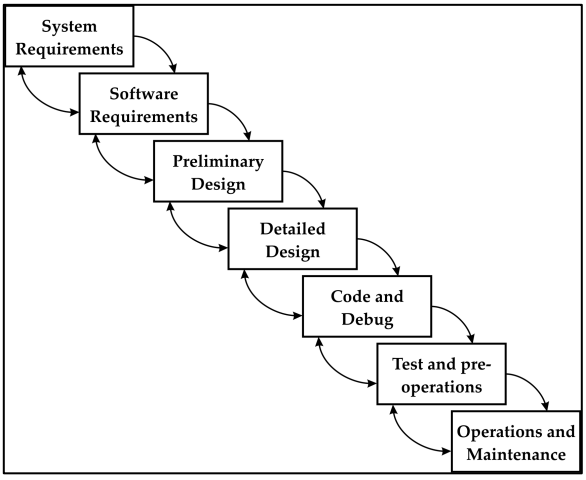

Figure 2: Waterfall model
Source: (Ehlers, 2011)

In his white paper (Royce, 1970) already indicated that the waterfall method is not sufficient. According to (Hoogendoorn, 2012) Royce is describing a software development method how it should not work. Royce supports a "do-it-twice" method, where software is delivered in smaller parts and where customer involvement is needed.

Criticism to the waterfall method is that it is slow and bureaucratic (Van der Klis, 2009) and it does not fit into the way of working of the developers nowadays (Larman & Basili, 2003). According to (Hoogendoorn, 2012) most of the projects have been executed based on the waterfall method (or variations on this model). When looking to those

projects, there are some key problems that can be identified (Hoogendoorn, 2012). Those key problems are visible in Table 2: Problems related to the waterfall method.

Problems in Waterfall Projects	Description
Knowledge disappears	Each phase is executed by a specific role, this role disappears when the phase is completed and therefor the knowledge. It is also not possible to document everything.
New Insights	Each phase will give the project team more information (new insights), but it is not possible to use this. All deliverables of previous phases completed.
Changing requirements	All requirements are completely elaborated in previous phases, so adding new requirements are inefficient and expensive.[1]
Complete	At the end of a phase the deliverable will not change anymore, so everything needs to be completed for 100%. A new requirement in a next phase is an imperfection of a previous phase. This will lead to delays in early phases.
To much documentation	In many cases there is too much documentation that it almost impossible to develop, imagine testing.
Estimation difficulties	Each phase requires its own activities and pace, this makes it hard to predict the lead-time needed for this phase (or activity).
Late risk's	Testing can only start when code is fully delivered. Solving errors will have higher costs when found in each later phase. The costs will increase exponentially.[2]

Table 2: Problems related to the waterfall method.
Source: (Hoogendoorn, 2012)

The waterfall method is a method that is sufficient for extending a known solution (Brooks, 2012). If the waterfall method is giving many problems, why is it still being used in software development projects? The waterfall method is an easy to use mode and is easy to explain, the waterfall method gives the impression that it is predictable and measurable and people are conservative and do not want to change (Hoogendoorn, 2012). Hoogendoorn (2012) continues that the waterfall method is a method that will not work and never have worked. The method should change from waterfall to iterative. Interesting from the initial study of (Royce, 1970) is that he actually refers to an iterative process (Hoogendoorn, 2012) and that it is different than the original explanation of the waterfall model (Ehlers,

[1] On average 20 to 25 percentage of the requirements will change during a project (Hoogendoorn, 2012)
[2] Boehm's law, 1987

2011). Ehlers continues his criticism with the fact that the paper of Royce in most cases only is referenced than read. A different approach to develop software is by using an agile method. In the next paragraph there will be an explanation about the characteristics of the agile software development method.

2.2 AGILE SOFTWARE DEVELOPMENT

The most traditional way of software development, as discussed before, is using the waterfall method. For several decades both developers and users have spent significant amounts of time to improve this methods (Huo et al. (2004). One of the major innovations in software development methodologies of the last few years has been the introduction of agile principles (Vlaaderen et al. (2011). Several companies, from small to large multinationals have adopted the agile methods (Dingsøyr et al. (2006). Previous studies have shown that agile software development methods can lead to significant benefits (Mann & Mauer, 2005) and (Versionone, 2008).

Agile software development finds it origin in 2001 when a manifest was published by a group of software practitioners and consultants (Abrahamsson et al. (2002). This manifest is called the: *Agile Software Development Manifesto* and contains four core values for the agile community. Those core values are[3]:

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

Next to four core values, the agile manifesto also contains twelve principles, which are visible in Table 3: Principles of Agile software development.

[3] http://agilemanifesto.org

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

Table 3: Principles of Agile software development
Source: http://agilemanifesto.org/principles.html

As mentioned earlier, agile is different from the traditional way of software development, but what is agile actually and especially *agile software development*? According to (Boehm, 2002) agile is a reaction against traditional methodologies, also known as rigorous or plan-driven software development methodologies. Looking to the methodologies we see that there are different methods, with all their own advantages or disadvantages. The six most common (Van der Klis, 2009) agile methodologies are: Extreme Programming, Crystal Methods, Scrum, Rational Unified Process (RUP), Dynamic Systems Development Method (DSDM) and Adaptive Software Development

(ASD). In Appendix A: Agile methodologies, there is a description of each of the six methodologies, based on (Abrahamsson et al. (2002).

This research is not a comparison between agile development methods versus the traditional development methods, although it is good to have a look to the differences between those two ways of software development. Both agile software development and the waterfall (traditional) method have their advantages and disadvantages. In Table 4: Comparison of Agile versus Traditional a comparison is displayed of the differences between agile methods and traditional methods (Awad, 2005).

	Agile	**Traditional / Heavyweight[4]**
Approach	Adaptive	Predictive
Success Measurement	Business value	Conformation to plan
Project Size	Small	Large
Management Style	Decentralized	Autocratic
Perspective to change	Change adaptability	Change sustainability
Culture	Leadership-Collaboration	Command-control
Documentation	Low	Heavy
Emphasis	People-oriented	Process-oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Upfront planning	Minimal	Comprehensive
Return on investment	Early in project	End of project
Team size	Small/creative	Large

Table 4: Comparison of Agile versus Traditional
Source: (Awad, 2005)

Agile software development has been a topic for literature and debates for several periods. Yet, academic research on this theme is rare (Abrahamsson et al. (2003). Most of the publications are written by practitioners and consultants (Abrahamsson et al. (2003).

[4] The waterfall method is a so-called heavyweight methodology (Awad, 2005)

Still it is good to have a look to the authors who have written about agile software development. The research of Van der Klis (2009) shows an overview of the studies related to agile software development in comparison with the Agile Manifesto, which are the principles of agile software development. In Table 5: Summarize of agile visions related to the agile principles, a summary of the agile visions, per main author is visible in relation to the agile principles of the Agile Manifesto.

Agile Manifesto (2001)	**(Miller, 2001)**	**(Highsmith & Cockburn, 2001)**	**Abrahamsson et al. (2003)**	**Boehm (2005)**	**Turner (2007)**
Customer satisfaction					Focus on value to customer
Accept changes	Adaptive	Continuous Improvement	Adaptive	Embracing change	Neutrality to change
Iterative	Iterative, time bound	Rapid delivery, Test constantly	Incremental	Fast cycles, frequent delivery, Refactoring	Short iterations delivering value
Cooperation	Collaborative		Cooperative	Pair programming	Team ownership
Motivation	People oriented				
Direct communication	Collaborative		Cooperative	Tacit Knowledge	
Working software	Incremental	Rapid delivery	Incremental		Continuous integration, Focus on value to customer
Sustainability		Rapid delivery	Incremental		
Continuous improvement					
Simplicity	Parsimony	Simplicity	Straightforward	Simple design	Lean attitude
Self-organizing teams					Team ownership
Regular reflecting	Modularity,	Continuous improvement	Straightforward	Retrospective or reflection	Learning attitude
Risk (Van der Klis, 2009)		Rapid delivery, Test constantly		Test-driven development	Test-driven

Table 5: Summarize of agile visions related to the agile principles
Source: (Van der Klis, 2009)

In his research, Van der Klis (2009), mentioned that risk is not a topic of the agile manifesto at all, while some authors have written about this. This item added by Van der Klis (2009) to the summary of the several visions on agile. After looking into those visions it is interesting to see that the principle: "Continuous improvement" is never mentioned and that only one author mentions the principle "Customer satisfaction".

Throughout this research we take into account the positive effects (Versionone, 2008) of using an agile software development methods comparing to the traditional way of software development. Although there are also other studies which have shown no significant difference between agile processes and structured processes (Estler (2012). Furthermore there are some factors that affect the selection of a development method. Those factors are project size, people and risk (Awad, 2005). This means that in some cases the traditional waterfall method can be the best method for the development.

2.2.1. Characteristics of Agile Software development

Interesting is to understand when a project is agile and how to define or measure it. One of the characteristics of agile software development is having a short iteration. The length of iterations can be influenced by several factors (Hoogendoorn, 2012). It can depend on the size of a project or release, if the team is new, changing of the priorities; focus can change when iterations are too long, etc. (Hoogendoorn, 2012). Several authors have mentioned that the maximum amount of weeks, of an iteration, should be around six weeks (Highsmith & Cockburn, 2001), (Abrahamsson et al. (2002) and (Miller, 2001). Another important characteristic is team interaction and especially the size of the team. The authors (Cockburn & Highsmith, 2001) say that large teams can make agile development more difficult. They continue that the average size of a team is nine people. The agile teams are small and compact. When the teams are bigger the efficiency will decrease and the

communication will increase (Hoogendoorn, 2012). According to Hoogendoorn (2012) the average size of should be is between eight and ten members. Better is to have smaller teams, but in projects which are complex with different technologies it is hard to have smaller teams (Hoogendoorn, 2012).

The authors (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) have given answer to the question: "*What makes a development method an agile one?*" They conclude that the development method should be:

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

Source: (Abrahamsson, Salo, Ronkainen, & Warsta, 2002)

2.2.2. Levels of agile

In the literature there is not a lot written about levels of agile. When answering an email on 21 May 2013, Dr. Pekka Abrahamsson explained that it is hard to measure the level of agile. Agile is a relative concept and therefore cannot be measured as such (P. Abrahamsson 2013, pers. comm., 21 May). One of the very limited studies about the level of agile is the doctoral dissertation of (Sidky, 2007). In the study of Sidky (2007) the level of agile is described in the following way:

> 'A set of agile practices that are related and, when adopted collectively, make significant improvements in the software development process, thereby leading to the realization of a core value of agility.'

The level of agile is the way of working of a company or within a certain project. Depending on the project the methods or agile methods can differ, but within the project the same methods should be used. According to (Hoogendoorn, 2012) multidisciplinary collaboration is essential for agile and therefore it should not be the case that team members use a different method.

One of the biggest challenges is to define the levels of agility (Sidky, 2007). It cannot be the case that the usage of an agile software development method will result in a different level of agile (Sidky, 2007). According to Sidky (2007) the levels must be based on the core values and qualities of agility. To do this the basis is the Agile Manifesto with the core values and principles that are described above. Sidky (2007) used the Agile Manifesto to limit the 12 principles into five values and qualities of agility. Those values and qualities are visible in: Table 5: Level of agile, together with the levels of agile. Sidky (2007) have defined five levels of agile, but it can be the case that a concept or component is available in level one and not in level five. This would make it a model, were not each component will increase when the agile level is higher. Each aspect should be measured in for each level and that is in the model of Sidky (2007) not the case. This is the reason the model is not fully adopted.

Agile level	Level Name	Agile Value or Quality
Level 5	Encompassing	Providing an all-encompassing agile environment
Level 4	Adaptive	Responding to change through multiple levels of feedback
Level 3	Effective	Producing quality software
Level 2	Evolutionary	Ensuring continuous delivery of software
Level 1	Collaborative	Establishing communication and collaboration

Table 5: Level of agile
Source: (Sidky, 2007)

2.3 SOFTWARE DEVELOPMENT QUALITY

An important part of a software development project is the quality of software. When measured according to the agreement with the project team and with the customer it will give information of the progress of the project or the actual quality of the software and development process itself. The problem about software quality is how to define it, how to measure it and when to measure it. What is the meaning of e.g. 1000 defects[5] in the software? And what means 1 defect after 10 test rounds and defect fixing? According to (Kitchenham & Pfleeger, 1996) "*software quality is a complex concept and means different things to different people*" (p.21). Further several authors have mentioned that there is no universal definition for software quality (Kitchenham & Pfleeger, 1996), (Phan, George, & Vogel, 1995), (Weinberg, 1991), (van Solingen & Berghout, 1999), (Jørgensen, 1999) and (Garvin, 1986).

One of the major problems during a software development project, as mentioned above, is defining of the quality of the software and especially how to measure this. Will it be the total amount of defects, the amount of defects per lines of code or the final opinion of the customer: the software is good or bad? According to (van Solingen & Berghout, 1999) software quality is having two fundamental complications: 1) It is hard to indicate the software quality in measurable terms and 2) it is hard to select the best development process to achieve the specified level of quality. Also (Phan, George, & Vogel, 1995) mentioned the problem of defining the quality of software. This is because of the immaturity of software quality. According to (Weinberg, 1991)"Quality is relative. What quality is to one person may even be lack of quality to another". Again quality, in a more general definition, is hard to define and hard to measure it, but the problem is also that managers are failing to communicate what exactly is meant with quality (Garvin, 1986).

[5] Common terminology about errors or bugs in a software application.

There are several moments during the process of software development when the quality is measured. It is done at the beginning of the project (how do we communicate it?), during the project and at the end by the approval of the final customer. During this study we do not look to the quality of the software based on the judgment of the customer. Although for the company it is important to know the opinion of the customer, but it can be a subjective opinion. We focus on the quality of the software during the development process itself and how it can be defined and measured.

According to the (ISO 25010:2010(E), 2010), the general definition of quality is: "The ability of a set of intrinsic characteristics to satisfy requirements". When we look more in depth to the quality of software (ISO 25010:2010(E), 2010) the ISO standard uses two main quality models: 1) Product quality and 2) Quality in use. Those two quality models are divided in 13 characteristics Table 6: Characteristic of software quality. The problem related to this is that the ISO standard is used for developed and deployed software and not about the process until the software is deployed and used. Although the ISO standard shows characteristics of the product quality and quality in use it does not describe the quality (and measurements) for the software development process itself.

Quality Model	Characteristic
Product Quality	Functional suitability
	Performance efficiency
	Compatibility
	Usability
	Reliability
	Security
	Maintainability
	Portability
Quality in use	Effectiveness
	Efficiency
	Satisfaction
	Freedom from risk
	Context coverage

Table 6: Characteristic of software quality
Source: (ISO 25010:2010(E), 2010)

During the development of software the project team want to measure the amount of errors, although we still do not know if 1000 defects is good or bad. Throughout the development process of the software the project team wants to measure the quality of the developed software, which is also described as "code". The less errors, the less time (and money) it cost to develop the software. When the quality of software will be measured after the development (ISO 25010:2010(E), 2010) based on characteristics or based on the customers judgment, how can it be measured during the development? Therefore we need to look into the process of the project generally. The software development project in general will look if the project is on time, within budget and according to the specifications (scope*)* (Toor & Ogunlana, 2010). According to (Cooke-Davies, 2002) there is a distinction between the success of the project and the project management success.

According to (Garvin, 1986) quality in general is so complex that it can be divided in five different perspectives. Those perspectives are visible in Table 7: Perspectives of quality and have been translated to software quality by (Kitchenham & Pfleeger, 1996).

Quality Perspectives	**Definition**
Transcendental view	Sees quality as something that can be recognized but not defined. This is on a more abstract level.
User view	Sees quality for purpose. Does the product meet the user's needs?
Manufacturing view	Sees quality as conformance to specification. This is an examination if the product was constructed "right the first time".
Product view	Sees quality as tied to inherent characteristics of the product.
Value-based view	*Sees quality as dependent on the amount a customer is willing to pay for it. In other words value for money.*

Table 7: Perspectives of quality
(Source: (Kitchenham & Pfleeger, 1996))

Looking to the earlier described five perspectives the "Manufacturing Perspective" is looking to the internal process (software development) itself. This view observes if the product was developed correct the first time. This means avoiding extra costs related to rework during development and after delivery (Kitchenham & Pfleeger, 1996).

When looking in to the Manufactures view of quality and measure this view, there are two suggested characteristics: *defect counts* and *rework costs* (Table 8: Characteristics manufactures view). Rework can lead to a lower overall quality of the software and by reducing the rework the product quality will improve (Deephouse, Mukhopadhyay, Goldenson, & Kellner, 1995). Measuring the rework in activities or costs will give a good insight of the software development process. A defect is one of the seven wastes of software development (Poppendieck & Poppendieck, 2003). They continue to say that a defect needs to be reduced as soon as possible, to minimize the impact.

Quality Perspectives	Definition
Defect counts	Number of known defects recorded against a product during development and use
Rework costs	Rework is defined as any additional effort required to find and fix problems after documents and code are formally signed-off as part of configuration management.

Table 8: Characteristics manufactures view
Source: (Kitchenham & Pfleeger, 1996)

2.4 (SOFTWARE) PROJECT MANAGEMENT

The focus for this research will be on software development projects. Therefore it is important to look into the key elements of Project Management and in specific Project Management for software development projects. This chapter will describe the main components of Project Management.

If the usage of an agile software development method is increasing the quality of the software development project, how should this be defined? When the success of a project in general need to be measured it is important to understand how *success* is defined. A project in general can be measured by common criteria like cost, time and quality (Atkinson, 1999); (Jugdev & Muller, 2005) and (Procaccino et al. (2005). Those criteria are also known as the *Iron Triangle* (Atkinson, 1999), Figure 3: Iron Triangle. The "Quality" aspect as a part of the Iron Triangle can also been seen as: "*according to specifications"* (Toor & Ogunlana, 2010) and it is not the quality of the product (software, house, car, etc.) itself. "*According to specification*" can also be mentioned as the "*Scope*" of a project. During this research we will use the terminology: Scope. The Iron Triangle is an extensively recognized measurement during last couple of periods (Toor & Ogunlana, 2010). A change to one of the components will lead automatically to a change of the other components. E.g when a project need to be finished earlier the cost and the scope will be affected. The costs will be less, but also the scope will be less. If the scope needs to be the same, the costs will increase (e.g. extra resources are needed).

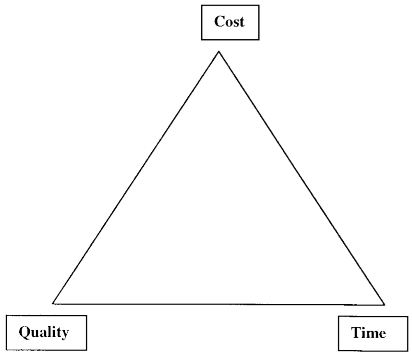

Figure 3: Iron Triangle

There is a difference between "*Project Success"* and "*Project Management Success* (de Wit, 1988). Project Management Success is measured based on the performance indicators as mentioned before: cost, time & quality, but that *Project Success* is measured based on the general objectives of a project (de Wit, 1988). According to (Cooke-Davies, 2002) project management success is the difference between success criteria and success factors. As described there is a difference between the success of a project and the success of project management. The project can be a success while the project management indicators are a failure or vice versa e.g. the project was not in budget or not on time.

> '*For example, the Sydney Opera House took 15 years to build and was 14 times over budget, yet it is proudly displayed as an engineering masterpiece*.'
> (Source: (Jugdev & Muller, 2005))

For this research the focus will be on the project itself, without looking to the external criteria. Some parties can have a subjective opinion about the outcome of a (software development) project while the activities went according to plan. During this study the focus will be on the internal activity. This means we will look into the criteria for project management success. "Good project management can contribute towards project success, it is unlikely to be able to prevent failure" (de Wit, 1988) p. 164). This means that the whole project team cannot prevent the final judgment of the stakeholders. If the project management was a success, the project still can be a failure. The proposition is that the

Cost, Time and Scope (software functionality and quality combined) are the key parameters to manage the process adopted in a project and therefore also the software development projects. There are several criteria that can be measured (Wateridge, 1998) and (de Wit, 1988). The most focus during projects will be on cost, time and scope (Atkinson, 1999). Based on the work from to (Karlesky & Vander Voord, 2008) the traditional project management is insufficient to manage the inevitable changes that are characteristic for software projects. They continue to say that agile project management is good to assist software development teams (and project managers) by managing the risks, scope, budget and planning of the project. This also confirmed by (Cockburn & Highsmith, 2001). They say that an agile methodology a better business performance, customer satisfaction and quality.

We can conclude, for this part of the review that cost, time and scope are the most important part of the project management indicators for the internal assessment of the project. There are more indicators for the success of a project, but those external indicators like customer satisfaction, end user satisfaction, etc. Those are all external indicators and during this research we are looking to the internal process. The question will be if an agile method will have a positive effect on the internal project management measurements.

2.5 THEORY AND PROPOSITIONS

This paragraph summarizes the literature review and describes the deduction of the theory, the associated propositions and conceptual model. Based on the theory, several probabilistic propositions are formulated for each aspect of software development projects. The formulated propositions are probabilistic, because it is assumed that the independent and dependent concepts on average increase or decrease on the same time.

When looking to the independent concept: level of agile working method, we can identify multiple characteristics about performing an agile project and how to identify when a project is agile (Highsmith & Cockburn, 2001), (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) and (Hoogendoorn, 2012). In Table 9: Characteristics & metrics of an agile method the characteristics and the metrics are visible related to performing an agile project.

Characteristics	Metrics
Iterations	*Maximum to six weeks*
Team size	*What is the teams size, up to ten members*
Adaptive	*Can the customer make changes?*
Cooperative	*How often is there contact between the customer and the developers*
Straightforward	*How easy is it to use the agile method*

Table 9: Characteristics & metrics of an agile method

The literature describes that it is very hard or difficult to describe software quality. The work of (Kitchenham & Pfleeger, 1996) and (Deephouse, Mukhopadhyay, Goldenson, & Kellner, 1995) describes that the important parts of measuring the software quality is using the characteristics number of defects and rework cost. By using an agile method it should reduce the number of defects and lower the rework costs. This all will lead to the following proposition.

- *P1: When the level of the "agile" method is higher, then it is likely that the quality of the software higher.*

The work of (de Wit, 1988) *and* (Cooke-Davies, 2002) *shows that important measurements for project management success are costs, time and scope. The expectation is that the level of agile has a direct, positive impact on those elements during when a project is done based on an agile method. Based on this theory the following three probabilistic propositions are made.*

- *P2: When the level of the "agile method is higher, then it is likely that the project will finish more closely to the estimated budget.*
- *P3: When the level of the "agile" method is higher, then it is likely that the project will finish more closely to the estimated deadline (planning).*
- *P4: When the level of the "agile" method is higher, then it is likely that the project will finish more accurate within in the defined scope.*

There has been made a choice to distinguish the three components of the iron triangle (Atkinson, 1999). *It also could be possible to make a proposition for a better project management result without defining it per individual item, but it makes it more interesting to see if there is a difference in the results when the aspects of the iron triangle are analysed as individual aspect.*

The above exploration leads towards the following conceptual model:

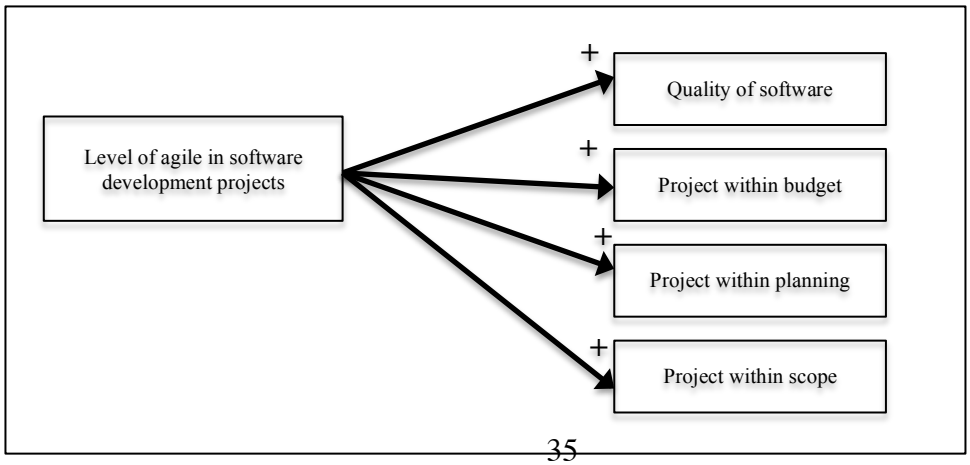

Figure 4: Conceptual model

The theory describes a simple causal relation between the independent concepts and the multiple dependent concepts. The level of agile in software development projects can lead to better quality of the software, projects can finish more accurate in budget, projects can finish more accurate in planning and projects can finish more accurate within scope. The objects of study for this research are projects. The specific research objective for this study is theory testing. The domain of the theory for which the propositions are believed to be true are all projects.

Chapter 3: Methodology

This chapter first describes the research design that is applied for this research. This will give an explanation for selecting a survey as a research design. Second the process of the data analysis is described. This will include the selection of instances and the method how to analyze the data. The last paragraph of this chapter will conclude the validity and reliability in this study.

3.1 RESEARCH DESIGN

For this study the chosen research strategy is a survey. This strategy is chosen to test the probabilistic relation between the level of agile software development projects (independent concept) and the four dependents concepts: 1) quality of the software, 2) project finished within budget, 3) project finished within budget and 4) project finished within scope. An experiment is the preferred research strategy (Dul & Hak, 2008), but will be not be feasible due to the available time. The current research needs to be done within approximately six months. Developing software, normally, will take longer than six months, including design, describing specification, developing, testing, etc. The second reason for not using the experiment as a research study is because of testing a probabilistic relation. To generalize the outcome a large number of instances ("Large N") are needed. The best way to do is by using a survey a research strategy. In case an experiment is not possible, a survey as research strategy is the second best option. The scores gained from the online questionnaire are analysed in a quantitative method.

3.2 DATA COLLECTION

The data collection will be done via an online questionnaire. The online questionnaire is created via an online questionnaire tool[6]. The questionnaire contains 33 questions about general company information, agile, project management and software quality. A questionnaire is a so-called self-completion questionnaire and according to Bryman & Bell (2011) there are several advantages of a self-completion questionnaire. Those advantages are: cheaper to administer, quicker to administer, absence of interviewer effects, no interviewer variability and convenience for respondents. Besides the advantages mentioned, there also some disadvantages like: difficult to ask a lot of questions, greater risk of missing data, lower response rate, cannot collect additional data, etc. (Bryman & Bell, 2011) and (Korzilius, 2000).

3.2.1 Sample selection

To find a large number of instances for this research, which are executing agile software development projects, a specific group of people need to be found. To find those people LinkedIn[7] will be used. The questionnaire will be communicated via LinkedIn user groups that have the agile or agile software development as topic. People who are having a LinkedIn account can be a member of a user group. For this research the population are three different LinkedIn user groups. The user groups and number of members are visible in Table 10: LinkedIn user groups. The total number of members are: 74.840, but because people can be a member of multiple user groups, members can be a duplicate. This means that there are at least 43.780 unique peoples who can answer the questionnaire.

[6] https://www.enquetesmaken.com
[7] Social media platform for business

LinkedIn User Group	Members[8]	Sample Size[9]
Agile and Lean Software Development	43.780	380-381 respondents
Agile	24.732	377-379 respondents
Lean Agile Software Development Community	6.328	361-364 respondents

Table 10: LinkedIn user groups

Via the group sites of LinkedIn it is possible to start a "discussion" which are visible for all members. This discussion function will be used to inform users about the questionnaire. This discussion will contain information about the research with a link to the questionnaire itself. During a period of one month the data will be collected and a discussion will be started four times in total. This will be done to inform the members about the questionnaire at to have as many respondents as possible.

There will be the situation of non-response. It is very unlikely that 43.780 will answer the questionnaire. Based on the population, which is the user group on LinkedIn, the sample size can be calculated (Korzilius, 2000). For the user group with over 43.000 members the sample size should be between 380 and 381 respondents. Via the questionnaire tool (www.enquetesmaken.com) it is possible to analyse if people have answered the questionnaire multiple times or did not fully complete the questionnaire. This will be discussed during the data analysis.

3.3 DATA ANALYSIS

To understand if the level of Agile is having an effect on the project management indicators, it is important to measure all concepts. In the subparagraphs below there is an explanation how each of the concepts will be measured. After the answers are received the first step will be to check the completeness of the answers. Also will be checked if answers need to modify. By this we mean making the answers generic. E.g. if people have answered

[8] Number of members at date 25-05-2013
[9] (Korzilius, 2000)

US, America, USA, Unites States, etc. one generic answer will be chosen. This will be done for all answers were free text was applicable. Always the original answers will be visible in relation to the corrected answers, done by the researchers.

3.3.1 Level of Agile

The level of agile will be measured based on the characteristics of agile software development: 2.2.1. Characteristics of Agile Software development. For each characteristic there are several questions that could lead to the following answers: Strongly Disagree (1), Tend to disagree (2), Neither Agree nor Disagree (3), Tend to agree (4) or Strongly agree (5). For this a likert scale is used. The scores with 1 relate to low level of agile and a score of 5 relates to a high level of agile. The measured level of agile is the answer of the respondent. This could be the level of agile of company of executed projects. The result could be different if a colleague of the initial respondent would answer the survey. In some case the score will be giving during the data analysis. This will be done for the questions 7 and 10. The full survey is visible in Appendix B: Survey. For each level of agile the score per characteristics will be calculated and divided by the number of questions (per characteristics three or four questions are available). This will result in a score per agile characteristic. The four scores per agile characteristic will be summarized and divided by four. This will lead to a level of agile score between 1 (low) and 5 (high). In Table 11: Example agile level calculation an example of the calculation is visible. The final level of agile is rounded to a whole number to have a score of 1, 2, 3, 4 or 5.

Agile characteristics	Score	Number of questions	Characteristic score
Incremental	11	3	3,7
Cooperative	15	3	5,0
Adaptive	20	4	5,0
Straightforward	15	4	3,8
FINAL AGILE LEVEL			**4**

Table 11: Example agile level calculation

3.3.2 Project planning & budget

A part of the project management triangle are time (planning) and cost (budget). Both indicators will be measured in the easiest way. Was the project finished within planning and or budget? Yes or no. If the project was not finished within planning and/or budget a second question will be asked. This will be used to measure with how many months (planning) and with which value the budget was extended. This will be needed to calculate the extension of planning or budget in percentage. An answer of no will have score 1 and an answer of yes will have a score of 3.

3.3.5 Project scope

The score of the score will be measured in the following way. The score for the scope is a combination of two questions. First the percentage that is needed before starting the project (0% is a high score and 100% is a low score). Agile would suggest a low percentage of all requirements, because not everything can be defined at the beginning of the project. Second how much of the initial scope is finally implemented in the final product (0% is a high score and 100% is a low score). If all of the defined requirements is needed it would suggest that a change is not possible. The combination of the two scores will give the score for the project scope. This can be a score between 1 (low) and 5 (high). This will be done based on the percentages: 0% - 20% is score 1, 20% - 40% is score 2, etc. until 80% - 100% is score 5. This will be answered by questions 21 and 22 of the survey.

3.3.5 Software quality

The software quality will be measured based on one question. And is related to how much of the actual software is approved by the quality and assurance department (Q&A) after the first round of testing. The final score will be between 1 (low) and 5 (high) and is

done based on the percentages given as answer: 0% - 20% is score 1, 20% - 40% is score 2, etc. until 80% - 100% is score 5. This is question 32 of the survey.

3.4 VALIDITY AND RELIABILITY

3.4.1 Validity

Validity can be divided into measurement validity, internal validity, external validity and ecological validity. Validity refers to the issues of whether or not an indicator measures the devised a concept. The external validity is concerned if the results of the study can be used for a generalization outside the specific research context (Bryman & Bell, 2011). Because the context of the study is focussed on software development projects, the findings are primarily in this context.

3.4.1 Reliability

Reliability concerns the consistencies and stability of the findings of the research (van Burg, 2011). Will the researcher have the same results after repeating the study or will a different research have the same result by doing the same study. Reliability can be divided into internal and external reliability. Internal reliability concerns if several researchers agree on what they have concerned (Bryman & Bell, 2011). During this research there is only one researcher and no observation have taken place. One researcher will do the interpretation and the analysis of the findings of the questionnaire.

The external reliability concerns about the issue if the indicators will give the same results during repeatable application within the same conditions. It is hard to say if there is absolute reliability. Will a respondent give the same answers or will a different respondent give the same answers about the same project as a colleague, because the research is about the results of a software development project.

Chapter 4: Analysis & Results

This chapter describes the exploration and the results of the collected data from respondents who have answered the online questionnaire. The first part of the chapter will describe the data in a more general way and the following paragraphs will explain the data per item of the conceptual model.

4.1 DESCRIPTIVE

The questionnaire was online for a total of 19 days. At July 9th the questionnaire was closed. In total 39 people have responded to the questionnaire and 29 of those respondents have entered the questionnaire completely. For some parts of the analysis it is possible to use the not full answered questionnaires. People from 13 different countries answered the questionnaire. This is visible in Table 12: Respondents per country.

Country	Percentage of respondents
USA	51,7%
UK	6,9%
Argentina	6,9%
Netherlands	3,4%
Canada	3,4%
India	3,4%
Sweden	3,4%
Australia	3,4%
Finland	3,4%
Germany	3,4%
Italy	3,4%
Switzerland	3,4%
Zimbabwe	3,4%

Table 12: Respondents per country

Scrum is the most used agile method. Almost 76 percent of the respondents are using the Scrum method within their software development projects. About 17 percent of the respondents are using another (not part of the default answers) or a hybrid method. This hybrid method contains aspects of different agile method, but is not an "official" agile method. The used agile methods are visible in: Table 13: Used agile methods

Agile method	Percentage of respondents
Scrum	*75,9%*
XP (Extreme Programming)	3,4%
Crystal Methods	3,4%
Other / hybrid	17,2%

Table 13: Used agile methods

4.2 LEVEL OF AGILE

The analysis of the data shows that, based on full answered questionnaires, the characteristics of agility: 2.2.1. Characteristics of Agile Software development, can be described. 72,4 percent of the respondents is working according to an agile level that can be defined as Level 4. Because there is no answer that will fit in a Level 2 group the results show that over 96 percent is working according to an agile level of 3 or higher. The agile level is defined on there to four indicators. If more indicators were used it is likely that the level of agile was more spread. The full results are visible in Figure 5: Level of Agile results.

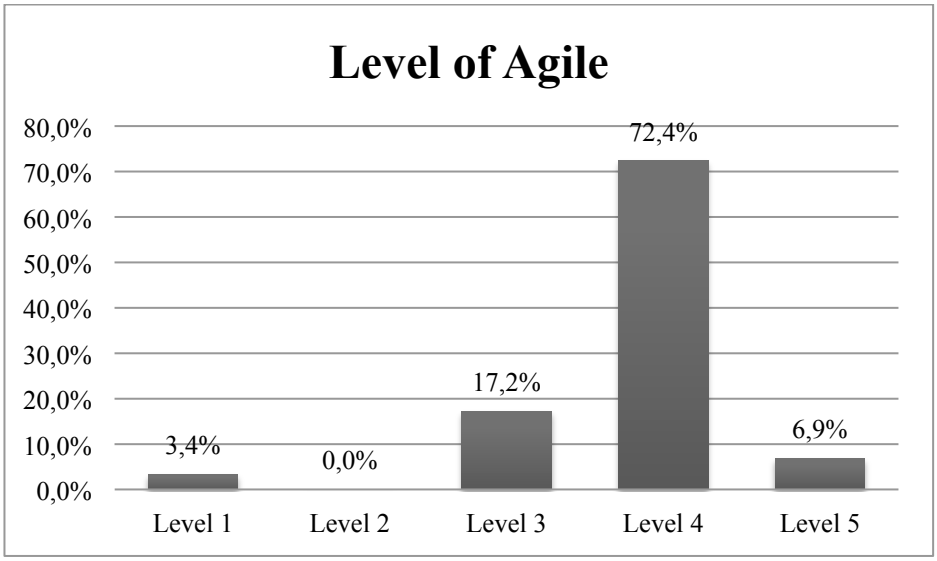

Figure 5: Level of Agile results

One of the questions was to give a score (from 0 to 10) about agility to the project/company of the respondent of the questionnaire. This means each level is having a range of two points. Level 1 will be between 0 and 2, Level 2 between 2 and 4, etc. It looks like that the respondents for level 3, 4 and 5 are having a realistic view on their agility level. Only for the respondents for level 1 give themselves the maximum score. 96 percent of the respondents give a score that is expected based on the level they fit in. The results are visible in Table 14: Level of agile & respondent agile score

Level of Agile	Score range	Agile score of respondents
Level 1	0-2	10,0
Level 2	2-4	0,0
Level 3	4-6	5,4
Level 4	6-8	6,8
Level 5	8-10	9,0

Table 14: Level of agile & respondent agile score

4.3 LEVEL OF AGILE VERSUS PROJECT ELEMENTS

As described before the level of agile will be tested against the four elements of managing a (software development) project. Those elements are planning, budget and quality and scope. In this paragraph each of those elements are discussed.

4.3.1 Level of Agile and planning

The relationship between level of agile and finishing the software development within project is illustrated in Figure 6: Level of Agile & Project Planning. In the chart the results of the respondents are visible. The score of the project planning is 1 (not within planning) or 3 (within planning). The result of R square is 0,0001. This means that 0,01 percent of the variation of achieving the initial deadline is explained by the model. Based on unreliable answers it is impossible to analyze the percentages of deviation between the initial planning and the actual execution of the project in relation with the level of agile. The results are visible in Table 15: Regression analysis of agile level and project planning.

	Model summary						**Parameter estimates**	
Equation	R	R Square	Adjusted R Square	Std. Error	F	Sig.	Constant	b1
Project Planning	*0,0100*	*0,0001*	*-0,0369*	*1,0357*	*0,0027*	*0,9589*	*1,9813*	*0,0140*

Table 15: Regression analysis of agile level and project planning

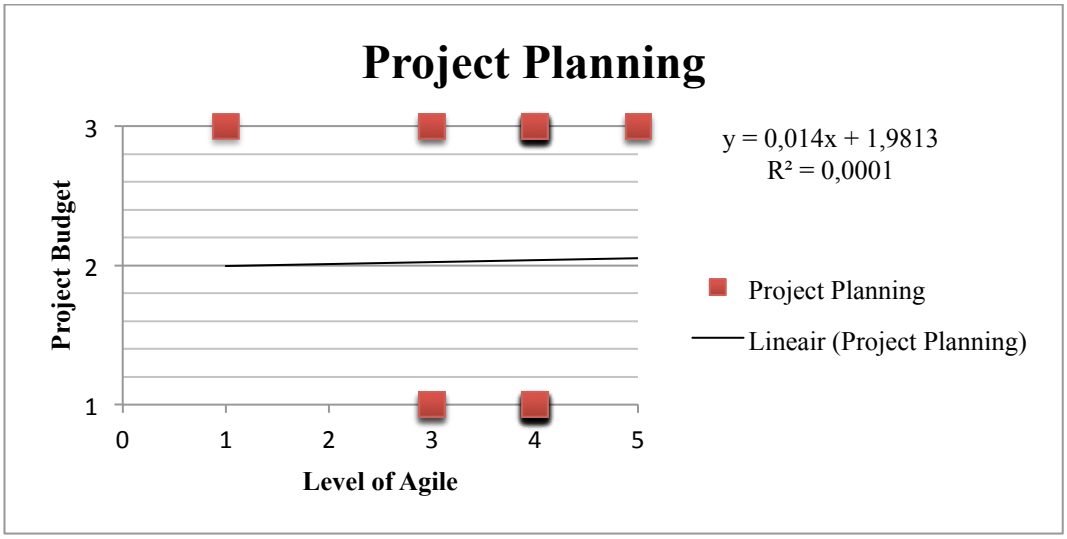

Figure 6: Level of Agile & Project Planning

4.3.2 Level of Agile and budget

It turns out that there is a negative effect on the executing a software development project within the initial budget when the level of agile will be higher. When the level of agile is higher than the software development projects will be executed less within the initial budget. The value of R Square is low. The model explains only 5,9 percent of the variation of finishing a software development within budget. There is no significant relation between these two variables. The p-value is 0,203. This means that the standard value of 5 percent is exceeded. The full results are visible in Figure 7: Level of Agile & Project budget. The score of the project planning is 1 (not within budget) or 3 (within budget). Based on unreliable answers it is impossible to analyze the percentages of deviation between the initial budget and the actual budget of the project in relation with the level of agile.

	Model summary						Parameter estimates	
Equation	R	R Square	Adjusted R Square	Std. Error	F	Sig.	Constant	b1
Project Budget	*0,244*	*0,0594*	*0,025*	*0,990*	*1,704*	*0,203*	*3,4486*	*-0,3364*

Table 16: Regression analysis of agile level and project budget

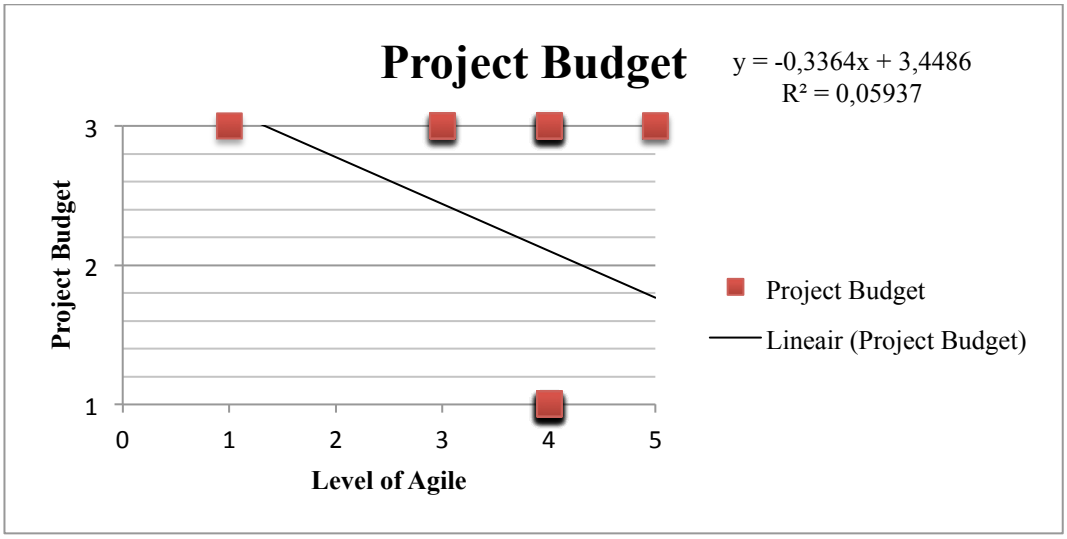

Figure 7: Level of Agile & Project budget

4.3.3 Level of Agile and quality

Like the budget aspect of a project there is also a negative effect between a higher agile level and the software quality of a software development project. The R square for this relationship is very low with 0,08 percent. The score for quality can be between 1 (Low) and 5 (high), which is visible in Table 17: Regression analysis of agile level and software quality. This based on the quality score which is given by the respondents. They could give a rate between 0-20%, 20-40%, etc. until 100%. The unknown scores are marked as zero percent and will therefore have the lowest quality score. The complete results are visible in Figure 8: Level of Agile & software quality. The p-value for the relationship between the level of agile and the software quality is 88,4 percent. This makes

the relationship not significant. It looks there no significant relationship between the level and agile and the software quality.

	Model summary						Parameter estimates	
Equation	R	R Square	Adjusted R Square	Std. Error	F	Sig.	Constant	b1
Project Quality	*0,0284*	*0,0008*	*-0,0362*	*1,4581*	*0,0218*	*0,8836*	*4,0748*	*-0,0561*

Table 17: Regression analysis of agile level and software quality

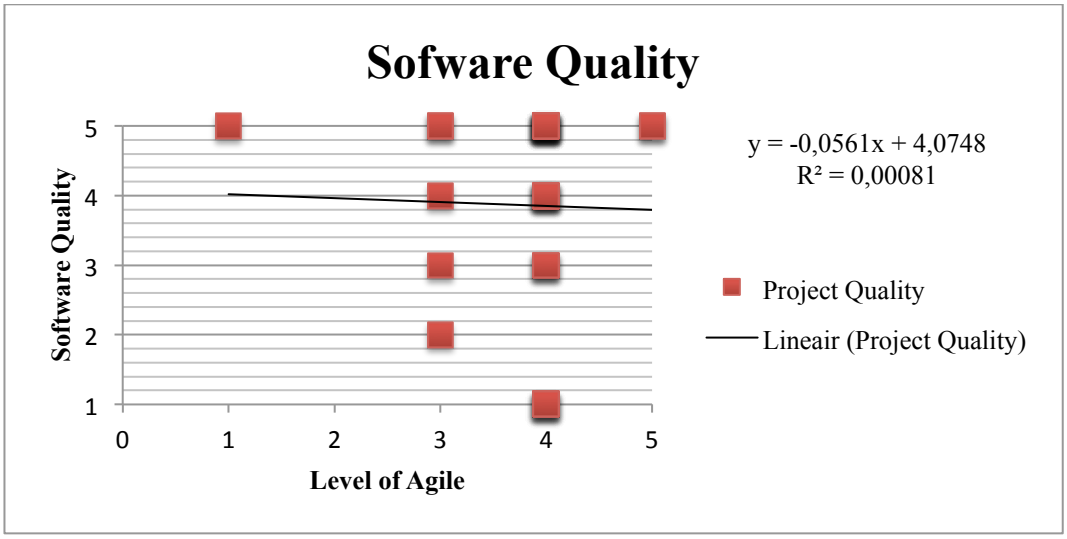

Figure 8: Level of Agile & software quality

4.3.4 Level of Agile and scope

Based on the results there is a negative effect between the level of agile and the scope of the project. The R square for this relationship is low with 1,6 percent. The score for scope can be between 1 (Low) and 5 (high) and is displayed in Figure 9: Level of Agile & Scope. The p-value for the relationship between the level of agile and the project scope is 34,4 percent. This makes this relation ship not significant. The results for this equation are visible in Table 18: Regression analysis of agile level and scope.

	Model summary						Parameter estimates	
Equation	R	R Square	Adjusted R Square	Std. Error	F	Sig.	Constant	b1
Project Scope	*0,1824*	*0,03328*	*-0,00205*	*1,2659*	*0,9294*	*0,3436*	*4,2103*	*-0,2827*

Table 18: Regression analysis of agile level and scope

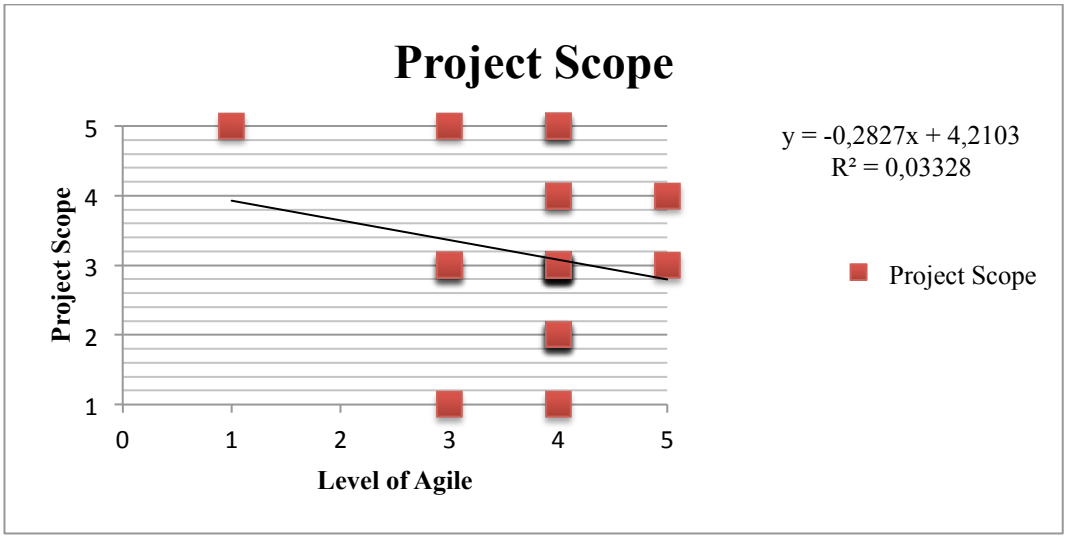

Figure 9: Level of Agile & Scope

Chapter 5: Conclusion & Discussion

This final chapter is arranged into four sections. The first part will be a discussion about this research. The chapter continues with the conclusions of the research. This chapter will be finalized with an advice for future research.

5.1 Discussion & Limitations

During this research a probabilistic relation is tested between the level of agile working methods in software development projects and the four dependents concepts of a project: 1) quality of the software, 2) project finished within budget, 3) project finished within budget and 4) project finished within scope. The expectation was to have a very high and positive relationship between the level of agile and the project characteristics. But the results show a complete different outcome.

As for almost all studies also this research is having limitations. Some limitations are well considered. The analysis of this research should be evaluated taking into account the following limitations. The project aspect of quality is measured based on the internal input from the respondents. Most of the respondents were not able to able to identify how the quality was measured, but they were able to give a score. Because the research was focused on the internal process the opinion of the customer was not taken into account. This could have led to other scores of quality.

Both the planning and budget aspect are only looking if the project was finished within the budget / planning. It gives no information how much the project was delayed or what the extra costs are. During the survey the questions was asked to identify the delay in the project and the extra costs, but an answer to this was not given. Therefore it was only possible to analyze if the project was within planning / budget. Although the results (of all

project aspects) are not significant a direction for both quality and scope are clear visible. This would require more in depth research to validate if this is correct.

Another limitation is the number of respondents. In total 39 people have answered the survey and in total 29 of those surveys are completely answered and used for the analysis. Based on 3.2.1 Sample selection a total of 380 respondents as part of the sample size was the goal. Only 7,6 percent of the expected sample size is achieved.

The last important limitation is related to the level of agile. As mentioned before, there is not a lot academic research available about the level of agile. 'Agility' is very abstract. The used definition, how to measure agile, is a method. There will be many more methods to measure the level of agile and this is one of the methods. This is a first step in the direction of measuring the level of agile, but further research is needed.

5.2 CONCLUSION

Based on the outcome we could say that there is for project aspects (budget, planning, scope and quality) that there is no significant relationship with the defined level of agile. Although the outcome is not significant some directions of the relations can be used. To summarize, this research has lead to the following main conclusions:

- There is no empirical evidence that a higher level of agile will lead to significant higher software quality;
- There is no empirical evidence that a higher level of agile will result in significant more often finishing within the initial planning and / or budget;
- There is no empirical evidence that a higher level of agile will result in significant more often finishing within the initial defined scope;
- The relation between higher level of agile and

Looking to the initial research question: "*Will the, level of the Agile working method, influence the quality of a project?*", we have to conclude that based on this research the total quality of a project will not significant be influenced by the level of agile.

5.3 FUTURE RESEARCH

During this research a first exploration was done related to the level of agile and relation of this to the defined project elements. Although the relationships are very low and the relationships are not significant, further analysis of level of agile would be advised. The level of agile is defined on a few indicators, but a more detailed clarification of the level of agile could be defined. The academic research, related to the level of agile is very limited. It is for software companies, very hard to say in which way they have adopted agility. The research of (Sidky, 2007) can be a very good starting point. It would be of added value if the level of agile can be defined including a method to measure this.

The project elements and especially planning and budget, are analyzed on a very high level; is the project in budget / planning yes or no? If the same relationship is tested a more detailed clarification of the element planning and budget would be needed. If a project is not finished within the initial planning / budget the interesting question would be with how many percent the project element was extended.

Finally it still would be interesting to repeat this research with the same research question by using a different method. For example by using a single or a multiple case study.

Chapter 6: References

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and Analysis. Espoo, Finland: VTT Publications.

Abrahamsson, P., Warsta, J., Siponen, M., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis . *Software Engineering, 2003. Proceedings. 25th International Conference on* , 244-254.

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management , 17* (6), 337-342.

Awad, M. (2005). A comparison between agile and traditional software development methodologies. University of Western Australia.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer , 35* (1), 64-69.

Brooks, C. (2012). Project Management and IT.

Bryman, A., & Bell, E. (2011). In *Business research methods*. USA: Oxford University Press.

Charette, R. N. (2005, September). *IEEE Spectrum*. Retrieved October 13, 2012, from http://spectrum.ieee.org/computing/software/why-software-fails

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

Cockburn, A., & Highsmith, J. (2001). Agile Software Development: The People Factor . *Computer , 34* (11), 131-133.

Cooke-Davies, T. (2002). The ''real'' success factors on projects. *International Journal of Project Management , 20* (3), 185-190.

Cugola, G., & Ghezzi, C. (1998). Software Processes: a Retrospective and a Path to the Future. *Software Process: Improvement and Practice , 4* (3), 101-123.

de Wit, A. (1988). Measurement of project success . *International Journal of Project Management , 6* (3), 164-170.

Deephouse, C., Mukhopadhyay, T., Goldenson, D., & Kellner, M. (1995). Software processes and project performance. *Journal of Management Information Systems , 12* (3), 187-205.

Dingsøyr, T., Hanssen, G., Dybå, T., & G. Anker, J. N. (2006). Developing software with scrum in small cross-organizational project. *Lecture notes in Computer Science* , 5-15.

Dul, J., & Hak, T. (2008). *Case Study Methodology in Business Research*. Oxford: Elsevier.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology , 50* (9-10), 833-859.

Ehlers, K. (2011). *Agile software development as managed sensemaking*. Doctoral dissertation, University of Stellenbosch, Stellenbosch.

Estler, H.-C., Nordio, M., Furia, C. A., Meyer, B., & Schneider, J. (2012). Agile vs. Structured Distributed Software Development: A Case Study. *2012 IEEE Seventh International Conference on Global Software Engineering* , 11-20.

Garvin, D. A. (1986). What does "Product Quality" Really Mean? *Sloan Management Review , 26* (1), 25-43.

Goldstein, H. (2005, September). *IEEE Spectrum*. Retrieved November 24, 2012, from http://spectrum.ieee.org/computing/software/who-killed-the-virtual-case-file

Highsmith, J., & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. *Computer , 34* (9), 120-122.

Hoogendoorn, S. (2012). In S. Hoogendoorn, *Dit is Agile*. Hoogendoorn, S. (2012). Dit is agile/druk 1. Pearson Education Benelux BV.

Huo, M., Verner, J., Zhu, L., & Babar, M. (2004). Software quality and agile methods. *28th Annual International Computer Software and Applications Conference* , 520-525.

IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology* . The Institute of Electrical and Electronics Engineers, New York.

ISO 25010:2010(E). (2010). *"Systems and Software Engineering—Systems and Software Product Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models,*. Geneva.

Jørgensen, M. (1999). Software quality measurement. *Advances in Engineering Software , 30* (12), 907-912.

Jugdev, K., & Muller, R. (2005). A retrospective look at our evolving understanding of project success. *Project Management Journal , 36* (4), 19-31.

Karlesky, M., & Vander Voord, M. (2008). Agile Project Management. *ESC 247, 267*. Boston.

Kitchenham, B., & Pfleeger, S. (1996). Software Quality: The Elusive Target. *Software, IEEE , 13* (1), 12-21.

Korzilius, H. (2000). De kern van survey-onderzoek. Uitgeverij van Gorcum.

Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer , 36* (6), 47-56.

Mann, C., & Mauer, F. (2005). A case study on the impact of scrum on overtime and customer satisfaction. *Agile Development Conference* , 70-79.

Miller, G. (2001). The Characteristics of Agile Software Processes. *The 39th International Conference of Object-Oriented Languages and Systems* (39).

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *The Journal of Systems and Software , 82* (9), 1749-1490.

Phan, D. D., George, J. F., & Vogel, D. R. (1995). Managing software quality in a very large development project. *Information & Management , 29* (5), 277-283.

Poppendieck, M., & Poppendieck, T. (2003). In *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional.

Procaccino, J., Verner, J., Shelfer, K., & Gefen, D. (2005). What do software practitioners really think about project success: an exploratory study. *Journal of Systems and Software , 78* (2), 194-203.

Royce, W. W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON* , 328-338.

Sidky, A. (2007). *A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework*. Doctoral dissertation, Virginia Polytechnic Institute and State University.

Standish Group International. (2010). *The CHAOS Report*. Standish Group International.

Toor, S.-u.-R., & Ogunlana, S. O. (2010). Beyond the 'iron triangle': Stakeholder perception of key performance indicators (KPIs) for large-scale public sector development projects. *International Journal of Project Management , 28* (3), 228-236.

van Burg, E. (2011). Kwaliteitscriteria voor ontwerpgericht wetenschappelijk onderzoek . In J. van Aken, & D. Andriessen, *Handboek ontwerpgericht wetenschappelijk onderzoek: Wetenschap met effect* (pp. 146-164). Boom Lemma uitgevers.

Van der Klis, P. (2009). *Agile software development bij een verzekeringsmaatschappij*. Master Thesis, Open Universiteit Nederland.

van Solingen, R., & Berghout, E. (1999). *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. Berkshire: McGraw-Hill Publishing Company.

Versionone. (2008). *The State of Agile Development*. Versionone.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology , 53* (1), 58-70.

Wateridge, J. (1998). How can IS/IT projects be measured for success? *International journal of Project Management , 16* (1), 59-63.

Weinberg, G. (1991). Quality software Management. In *Quality Software Management: Anticipating Change*. (p. 336). Dorset House.

Appendix A: Agile methodologies

In the table below there is a description of the six most common agile methods described by (Van der Klis, 2009). The table also shows per agile method the key points, special features and the shortcomings as mentioned by (Abrahamsson, Salo, Ronkainen, & Warsta, 2002).

Agile Method	Authors	Key points	Special features	Identified shortcomings
Extreme Programming	Highsmith (2002), Abrahamsson (2002), Abrahamsson (2003) Williams (2004) Strode (2005)	Customer driven development, small teams,	Refactoring – the ongoing redesign of the system to improve its performance and responsiveness to change.	While individual practices are suitable for many situations, overall view and management practices are given less attention.
Crystal Methods	Highsmith (2002), Abrahamsson (2002), Abrahamsson (2003) Williams (2004) Strode (2005)	Family of methods. Each has the same underlying core values and principles. Techniques, roles, tools and standards vary	Method design principles. Ability to select the most suitable method based on project size and criticality	Too early to estimate: only two of four suggested methods exist.
Scrum	Highsmith (2002), Abrahamsson (2002), Abrahamsson (2003) Williams (2004) Strode (2005)	Independent, small, self-organizing development teams, 30-day release cycles.	Enforce an paradigm shift from the 'defined an repeatable' to the 'new product development view Scrum'	While Scrum details in specific how to manage the 30-day release cycle, the integration and acceptance tests are not detailed.
Rational Unified Process (RUP)	Abrahamsson (2002),	Complete software development model, including tool support. Activity driven role assignment	Business modelling, tool family support	RUP has no limitations in the scope of use. A description how to tailor, in specific, to changing needs is missing
Dynamic Systems Development Method	Highsmith (2002), Abrahamsson (2002), Abrahamsson (2003) Strode (2005)	Application of controls to RAD, use of time boxing, empowered DSDM teams, active consortium to steer the method development	First truly agile software development method, use of prototyping, several user-roles: ambassador, visionary and advisor	While the method is available, only consortium members have access to white papers dealing with the actual use of the method.
Adaptive Software Development	Highsmith (2002), Abrahamsson (2002), Abrahamsson (2003) Strode (2005)	Adaptive culture, collaboration, mission driven component based iterative development	Organizations are seen as adaptive systems. Creating an emergent order out of a web of interconnected individuals.	ASD is more about concepts and culture than the software practice

Source: (Abrahamsson, Salo, Ronkainen, & Warsta, 2002)

Appendix B: Survey

Question ID	Question	Answer option
1	In which country is the headquarters of your company located?	Free answer
2	How much employees are working in your organization?	Free answer
3	Which method of agile do you use?	XP (Extreme Programming) Scrum DSDM (Dynamic Systems Development Method) Rational Unified Process (RUP) ASD (Adaptive Software Development) Crystal Methods Other
4	For how many years is the company using an Agile method?	Never Less than 6 months Between 6 and 12 months Between 1 and 2 years Between 2 and 5 years More than 5 years
5	What is the average number of team members in an agile software development project?	Between 0 - 5 members Between 5 - 10 members Between 10 - 15 members Between 15 - 20 members More than 20 members
6	What percent (%) of your company's software projects use an Agile methodology?	A value between 0 & 100
7	What is the time of an average iteration between two deliveries?	1 - 3 weeks 3 - 6 weeks 6 - 9 weeks 9 - 12 weeks More than 12 weeks
8	It is a common practice to divide the system up into mini-projects. The system is seldom developed as one large project.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
9	All the iterations and releases, including the requirements, are part of the plan for the whole project.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree

10	If a problem occurred that affected the schedule or requirements of a project, the client was updated.	When the delivery was done By the end of the week Immediately By the end of the day Never
11	The customer has the authority to decide what is being developed in which iteration.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
12	When the project manager was unreachable, at any point in time all team members had enough information to update the customer about the exact status of the project.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
13	In order to deliver valuable software to clients, change should be welcomed and not constrained.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
14	The plan for upcoming iteration may change based on customer feedback from the previous or current iteration.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
15	The customer is given the authority to direct what is being developed in which iteration.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
16	The customer can give his/her feedback throughout the development process even if it means that requirements must be changed.	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
17	Does the project needs to maintain a high process ceremony due to certain audits or regulations?	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
18	All our project plans are always documented?	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
19	The agile method we are using is easy to use?	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree

20	The agile method we are using is easy to learn?	Strongly Disagree Tend to disagree Neither Agree nor Disagree Tend to agree Strongly agree
21	We start a project (development) when the following percentage of requirements are available:	A value between 0 & 100
22	What percent (%) of the initial requirements is part of the final 'product'?	A value between 0 & 100
23	What was the initial planning of the project?	Free value
24	Was the project finished within the initial planning?	Yes No
25	With how many months the project was delayed?	Free value
26	What was the initial budget of the project?	Free value
27	Was the project finished within the initial budget?	Yes No
28	With how much is the initial budget increased?	Free value
29	How do you measure the software quality during the development phase?	Measuring the defects (bugs / errors) Calculating the rework costs Each delivery needs approval from the customer Q&A gives approval Other:
30	How good is the software quality after development (per delivery)?	Score between 1 & 10
31	What were the amount errors per 1000 lines of code on average per delivery?	Unknown Number of errors per 1000 lines of code (free value)
32	How much % of the developed software is passed Q&A (and good for the customer/delivery) after the first round of testing?	80 - 100% 60 - 80% 40 - 60% 20 - 40% 0 - 20% Unknown
33	If you could rate your company and the projects about the 'agility level' what would it be? Agility means: how agile are we	A value between 0 & 10