The Success of Agile Software Development

~ Graduation Thesis ~

Author : Raditeo Warma

Issue Date : June 8th, 2012

Graduation Thesis

Fontys University of Applied Sciences

HBO-ICT: English Stream

Data student:	
Family name , initials:	Warma, Raditeo
Student number:	2188207
project period: (from – till)	February – June 2012
Data company:	
Name company/institution:	The Lectorate Software Quality and Testing of Fontys ICT
Department:	
Address:	Rachelsmolen 1, 5600 AH Eindhoven, Building R1
Company tutor:	
Family name, initials:	Brunekreef, Jacob
Position:	Project Leader
University tutor:	
Family name , initials:	Zijlmans, Jack
Final report:	
Title:	The Success of Agile Software Development
Date:	June 7th, 2012

Approved and signed by the company tutor:

Date: June 8th, 2012

Signature:

Preface

This report was written for fulfilling requirements of graduation to finish my study and acquire a bachelor's degree at Information & Communication Technology department, Fontys University of Applied Sciences. This report is carried out as a result of my graduation project at lectorate Software Quality and Testing, Fontys ICT. The lectorate Software Quality and Testing has been participating in the EQuA project that has started in the fall of 2010. The title of my project is “The Success of Agile Software Development." The assignment of the project is to find theoretical and practical evidence of the success of agile software development.

In this project, the author was under supervision of Mr. Jacob Brunekreef as the company mentor and Mr. Jack Zijlmans as the university mentor. Thanks to all interviewees. Mr. Brian Teunissen and Mr. Patrick Verheij, they were consultants, that work at Inspearit B.V., Netherlands. Mr. Dody Muktiwibowo, he was a consultant, that works at Astra Graphia Information Technology, Indonesia. They were willing to spend their time to be interviewed as the resources for my research. For all web-survey-respondents, I say thank you for the participations in my web survey. Special thanks to Mr. Jacob Brunekreef for his kind assistances, guidance and advices.

Eindhoven, June 8th 2012

Raditeo Warma

Table of Contents

Summary

In the last decades, business community desires solutions that able to provide faster and better responses to change in software development. Many proposals regarding the improvement of software development methods have been suggested in the scope of standardization and measurement of tools, practices, techniques and software process.

Recently, many remedies for improvement have come from practitioners with years of experience. They called their methods as Agile Software Development. They claimed that it was possible to provide better software quality with higher customer satisfaction compared to traditional methods. This claim has made a big impact on the software development world. It was becoming a subject of debate among software practitioners for years. Some of the practitioners agreed with this claim. Some others are doubted. The rest flatly refused on this claim.

This project aims to find the proofs regarding the claim from scientific articles and practices in the field of work. During the project, the researcher gathered and analyzed 33 articles about the relevant topics. These articles were grouped into two groups. The first group, 30 articles, reviewed the success of agile software development. The second group, three articles, reviewed the adoption of agile software development. From the first group, the researcher gathered 29 articles including some comparative studies that support the above claim. One study that, according to the author, claimed that agile and traditional methods satisfy their respective customers under a wide range of different situations. In terms of company reports or company case-studies, agile and traditional methods cannot be compared with each other since each method was implemented in different projects with different initial conditions, people, and circumstance.

Furthermore, in order to gather data from practitioners of software development, the researcher interviewed three agile consultants from the Netherlands and Indonesia. The results of the interview provide subjective information regarding agile software development. In addition, the researcher ran a web survey consisting of 23 questions. The researcher invited more than 279 prospective participants from both the Netherlands and Indonesia. In the beginning, the researcher did not have a plan to utilize a web survey to collect data. Over the time, the project did not go according to project plan. Thus, to obtain data in a bulk in short time, the researcher ran this web survey instead of using interviews.

Glossary

AM	Agile Modeling
AMDD	Agile Model Driven Development
ASD	Adaptive Software Development
CASP	Critical Appraisal Skills Program
CRC	Class Responsibility Collaborator
DSDM	Dynamic Systems Development Method
EQuA	Early Quality Assurance in software production
ICT	Information and Communication Technology
JIT	Just-in-Time
OO	Object Oriented
QA	Quality Assurance
S1…S33	Study 1…Study 33
SDLC	Software Development Life Cycle
SQA	Software Quality Assurance
TDD	Test Driven Development
XP	Extreme Programming

1. INTRODUCTION

These days, software development has been increasing. Demand of software application in organizations and government agencies is increasing. This is because of the growing awareness of the organizations regarding importance of software application to help business processes. This software application would improve performance of the business process, speed up the business processes, and reduce costs that have to be paid by the organization.

In software development, there are some things that need to be noticed ranging from human resources that handles the project, up to what method that should be applied in the software development process. One of the software development methods is Agile Software Development.

The word “agile” means fast, lightweight, nimble, and alert. “Agile” was used as a term to describe the concept of a process model that is different from the concept of the process model that already exists. The concept of agile software development was coined by Kent Beck and sixteen colleagues by stating that agile software development is a way for building software by doing it and helping others build it all at once.

Agile software development with its characteristics those are fast development, lightweight processes, nimble and alert to changes was claimed as better development approach compared to the traditional approach that already exists like the waterfall model. Agile was claimed that it was able to provide higher software quality with higher satisfaction of the customer. This claim needs to be evidenced. The claim was the trigger of “The Success of Agile Software Development” project to prove the above claim.

All the topics are represented in seven chapters. Chapter 1 represents a brief introduction about agile software development and its relation to the assignment. Chapter 2 represents description of company profile where the assignment was run. Chapter 3 represents description of the assignment, initial condition and research questions. Chapter 4 represents introduction to the research. Chapter 5 represents introduction to the concepts of agile software development and briefly describes several agile methods. Chapter 6 provides findings of the research. Chapter 7 represents conclusions and recommendations. Additionally, attachments (literature summary, interview questionnaire, web survey questionnaire, and project plan) can be found in appendix.

2. COMPANY PROFILE

Fontys ICT is a leading and innovative Institute that provides education at Bachelor, Master, and Associated Degrees. Nearly 2000 students are taught everything regarding practice of ICT field. One of the lectorates of Fontys ICT is the lectorate Software Quality & Testing where I am working for. The lectorate Software Quality & Testing provides students a guideline in order to form practitioners who are competent in related knowledge and skills with software quality assurance (QA) and software testing.

From the fall of 2010, the lectorate Software Quality & Testing has been participating in the EQuA project. EQuA is an acronym for Early Quality Assurance in software production. The project aims at the detection and correction of errors in the process of software production, in a stage as early as possible. It stated otherwise: the project aims at achieving a quality standard as high as possible in a stage as early as possible.

Both in the scientific world and in the software-development community, the quality problem is addressed. The goal of the EQuA project is to bring together knowledge and insights from both sides, and from there to create practical solutions.

The EQuA project is a collaboration of eight partners: two universities of applied science, three scientific institutions and three IT companies (Fontys, Hogeschool van Amsterdam, CWI, Sogeti, TU Delft, Info Support, SIG, and TU/e). Fontys ICT manages the project.

3. ASSIGNMENT OVERVIEW

Agile software development is gaining interest from academia and industry. Although many articles and books have discussed agile software-development methods, few of them discussed the impacts of agile methods on software quality and customer satisfaction. Agile supporters claim that the use of agile methods leads to better software quality with higher customer satisfaction compared to the use of traditional methods. Importantly, evidence for this claim was needed. This was the starting point of this research, and resulted in the following research questions:

1. What theoretical and practical evidence can be found in the literature for the claims mentioned above?
2. What practical evidence can be found in IT companies applying agile software-development methods?
3. What selection factors of agile methods instead of traditional methods are influenced by cultural aspect?

This project aims to analyze information that has been obtained in order to answer the research questions. Project results are purposed to prove that agile is used and successfully works, and it is resulting in better results than traditional methods. The researcher is not going to examine the success and failure factors of the use of both methods.

4. RESEARCH

Agile software development methods were created to respond to the business world asking for fast, agile, and lightweight software-development techniques for anticipating the rapidly growing software industry. Agile claims that its methods provide better software quality with higher customer satisfaction compared to traditional methods such as the waterfall model.

The researcher is expected to find evidence for the above claim, and find sufficient information regarding agile adoption and the results of the use of agile methods in scientific literature and practical in software companies.

In order to do that, the researcher is going to use three gathering information methods: literature review, expert interviews, and a web survey. The researcher gathers scientific literature such as journal papers, reports, surveys, books, etc., which contains needed information. Furthermore, to find practical evidence in software companies, the researcher has to interview some agile experts and make a web survey that addresses experience of software development practitioners in the software development field.

By the interview and survey, the researcher tries to obtain the information regarding opinions and experience of the practitioner during working in the software-development project. Then the researcher analyzes the information from both sources to get the answers for the research question, and put the results in the report.

5. CONCEPTS

The word “agile” means fast, lightweight, nimble, and alert to changes. “Agile” was used as a term to describe the concept of the process model that is different from the concept of the process model that already exists. The agile software-development concept was coined by Kent Beck and sixteen colleagues by stating that agile software development is a way of building software by doing it and helping others build it all at once.

In agile software development, interaction and personnel are more important than processes and tools. Working software is more important than complete documentation. Collaboration with clients is more important than contract negotiation. Other than that, responsiveness to changes is more important than following plan. The agile software development process enables tolerance to the changing of requirements, so that the changes can be quickly addressed.

5.1 The Principle of Agile Software Development

One of the main characteristics of agile software development is the capability of team to respond to changes. Why? Because the changes are the main thing in software development such as changing requirements of software, changing team members, changing technology, etc. In addition, agile software development also emphasizes the importance of communication among the team members, between technical people and businessmen, between developer and manager. Another feature is the clients to be part of the development team. These characteristics are supported by 12 principles that have been set out by Agile Alliance. According to the Agile Alliance (2012), these principles are for those who want to succeed in application of agile software development:

1. Client satisfaction is the top priority by producing products early and continuous.
2. Accept the changes of requirements, even at the end of development.
3. Deliver products in weeks (2-8 weeks).
4. Business people and developers work together every day throughout the project.
5. Build software in the environment of people who are highly motivated.
6. Face-to-face communication is an effective and efficient communication.
7. Working software is the main measure of project progress.
8. Stable support of sponsors, developers, and users is required to maintain a sustainable development.
9. Attention to the technical intension and good design enhances nature of agile software development.
10. Simplicity
11. Good architecture, requirements, and design emerge from a self-organizing team.
12. Periodically, the team conducts self-evaluations and finds ways to be more effective and efficient.

The twelve principles have become the basis for models that have the nature of the agile process. These principles attempt to deal with three critical assumptions about typical software projects:

1. Software requirements are difficult to predict from beginning and always change. In addition, client priorities often change over the project.
2. Design and construction often overlap. It is difficult to estimate how far the design is required prior to the construction.
3. Analysis, design, development and testing cannot be predicted as desired.

5.2 Human Factor on Agile Methods

The key of human factors in this model is based on the needs of people and team, not the otherwise. To be able to be successfully implementing an agile process model, there are some important keys:

1. Competence: developer's skills in building and knowledge about the process of building.
2. Focus: each team member has the same focus even though they have a different role in the teams.
3. Collaboration: developers working with clients, other team members and managers.
4. Ability of decision making: the development team has autonomy to take decisions related to the project and technical issues.
5. Fuzzy problem-solving ability: the team is able to finish in sorting out the important issues to be solved immediately or later.
6. Mutual trust and respect: excellent teamwork is supported by a sense of trust and mutual respect one and another.
7. Self-management: the team sets itself up, manages the process to suit its environment, and schedules itself to deliver the results.

5.3 Agile Software-Development Methods

The above points reveal that agile software development is a software development approach that has a different concept from the traditional approach. It is a new concept that emphasizes human collaboration, response to changes, and results of working software. It overrides contract negotiation, following plan, and complete documentation that are regarded as waste. However, agile software-development methods still employ activities that are owned by traditional methods, but in a different way. Both agile and traditional methods employ Analysis, Design, Implementation, and Test (this series of activities is known as SDLC). The distinction is, in the agile way, SDLC is performed more frequently. This activity ensures software quality assurance (SQA) is performed more frequently. Normally, an SDLC is completed within an iteration. Figure 1 will give an idea how it happens.

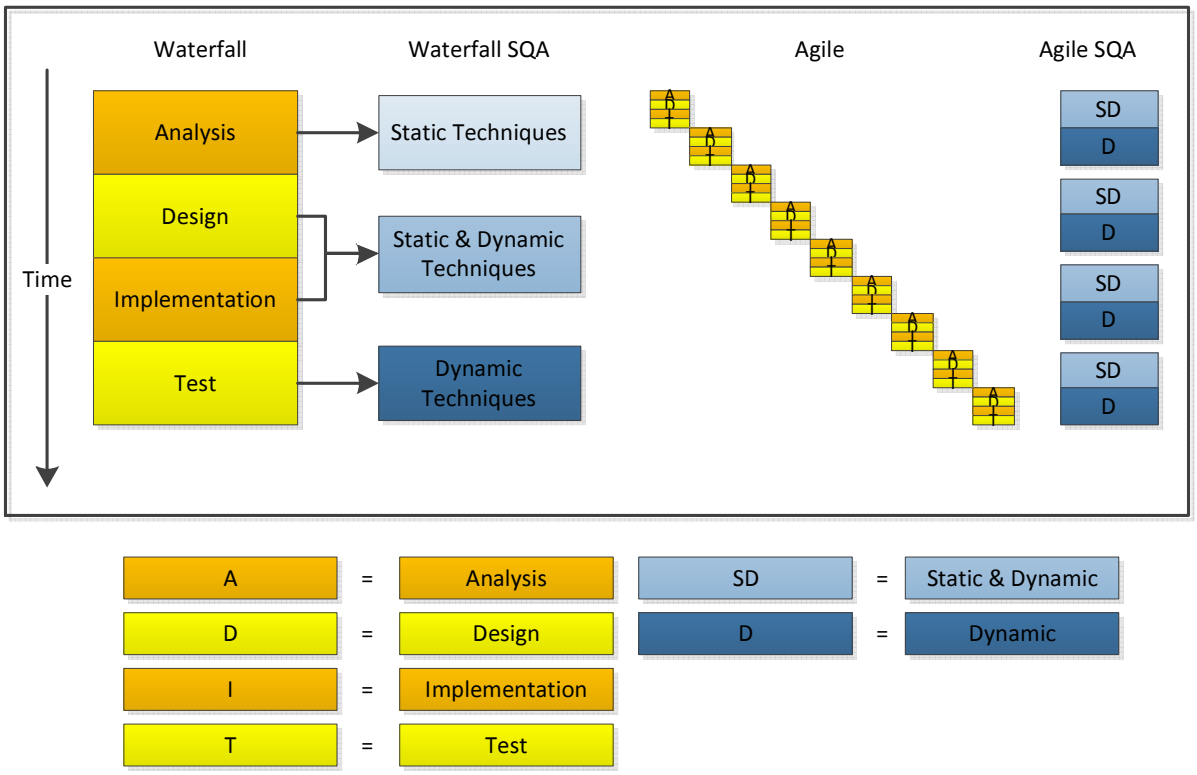


Figure 1. Software quality assurance timeline (Ming Huo et all, 2004)

Figure 2 gives an overview of a development process in iteration cycles until the product is ready to release to market.

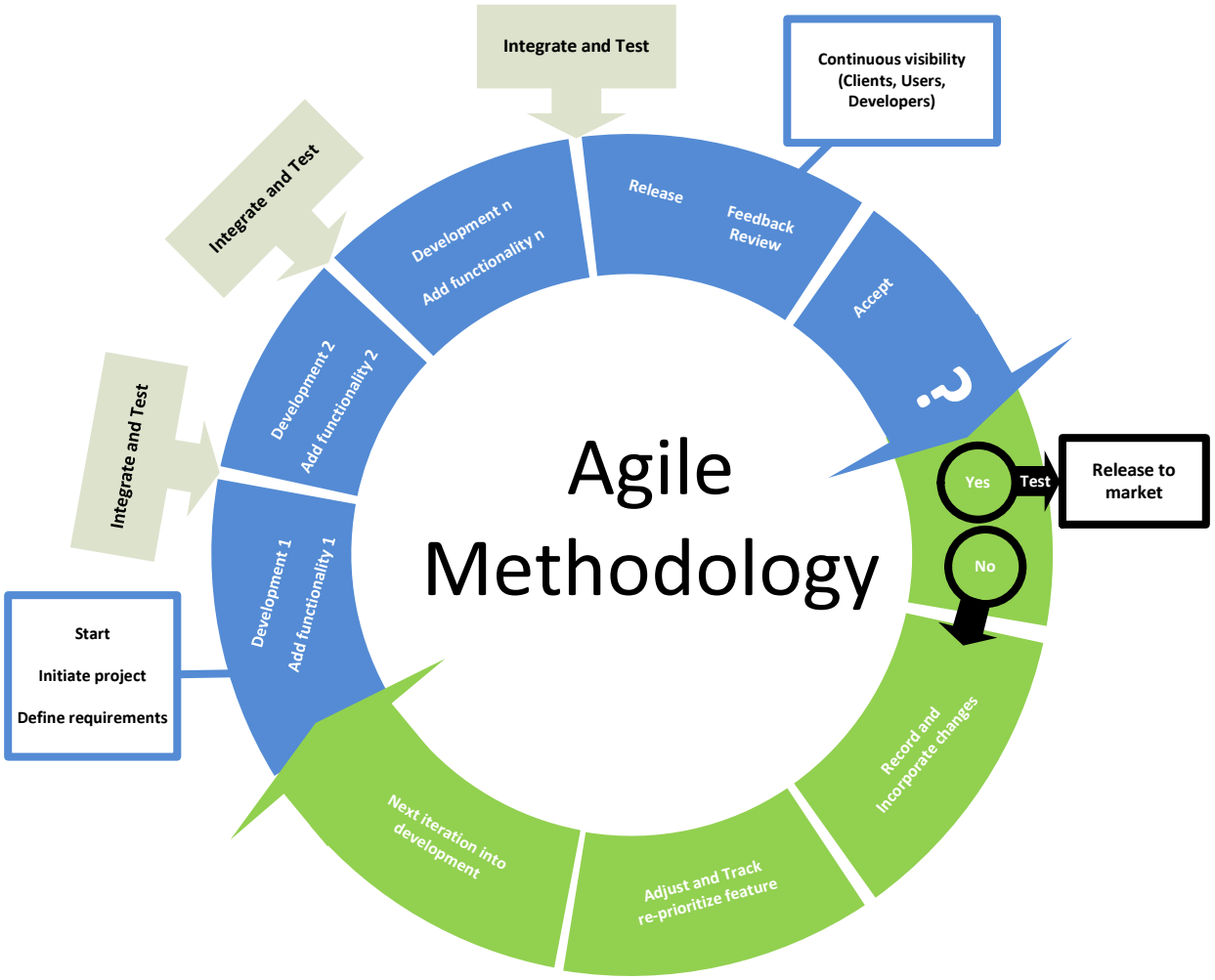


Figure 2. Iteration process of agile methodology

The below parts will explain several variants of agile software development methods, with aim to provide an overview how the methods work for the development process. The following methods are included in agile software-development methods:

5.3.1 Extreme Programming

Extreme Programming, known as XP, has been published by Kent Beck in 1999. XP is using the object-oriented approach in its process model. Planning activity in this method is gathering user stories from client in which the client sets the priorities. Each story sets prices and period of development. If too large, the story can be broken down into several smaller stories. XP has a principle in design activity that is simplicity. XP utilizes CRC cards (Class Responsibility Collaborator) to identify and organize the classes in OO concept. In case XP encounters difficulties, a prototype is built (this is named spike solution), then refactoring is performed, which is developing design of software after it has been written. Encoding activity includes preparation of unit tests prior to coding. This activity is used as the focus of

programmers to create software. Pair programming is done for real-time problem-solving and real-time QA (quality assurance). Furthermore, testing activity is using the unit tests that were prepared prior to encoding (2(Jeffries, 2003).

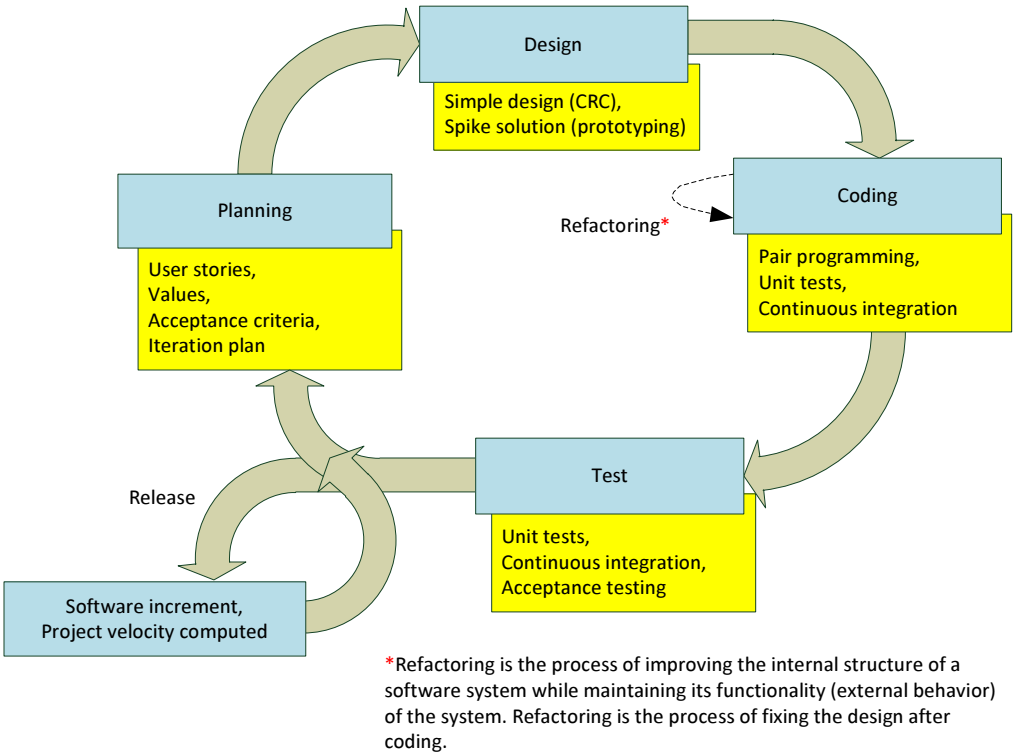


Figure 3. Workflow activity of extreme programming (Umi Proboyekti)

5.3.2 Scrum

Scrum has been introduced by Jeff Sutherland in the early of 1990s, and further development has been carried out by Schwaber and Beedle. There are principles that are emphasized by Scrum in its process model. The small size of the team, enables smooth communication, reduces costs, and empowers the team members. Development process can adapt to the changing technical and changing business issues. The process is generating software increments. People who develop software are divided into small teams. Documentation and testing continue to be done after the software has been built. In Scrum, developer is able to state that the product is finished whenever it is needed.

Scrum activities include Backlog, Sprints, Scrum Meetings, Demo. Backlog activity: Backlog is a list of requirements, prioritized by the client. This list can grow depending on the situation of the project and condition of the client. Sprint activity: units of work, that are required to meet the requirements that have been set out in the backlog within development time, are specified in a time-box (1-4 weeks). During this process, there is no new backlog addition. Scrum meeting activity: a meeting of 15 minutes every day at the beginning of the day to evaluate what was done, constraints, and target completion. Demo activity: delivery of software increment to the clients, the software increment is demonstrated in front of the client and evaluated by the client (2(Kniberg, 2007).

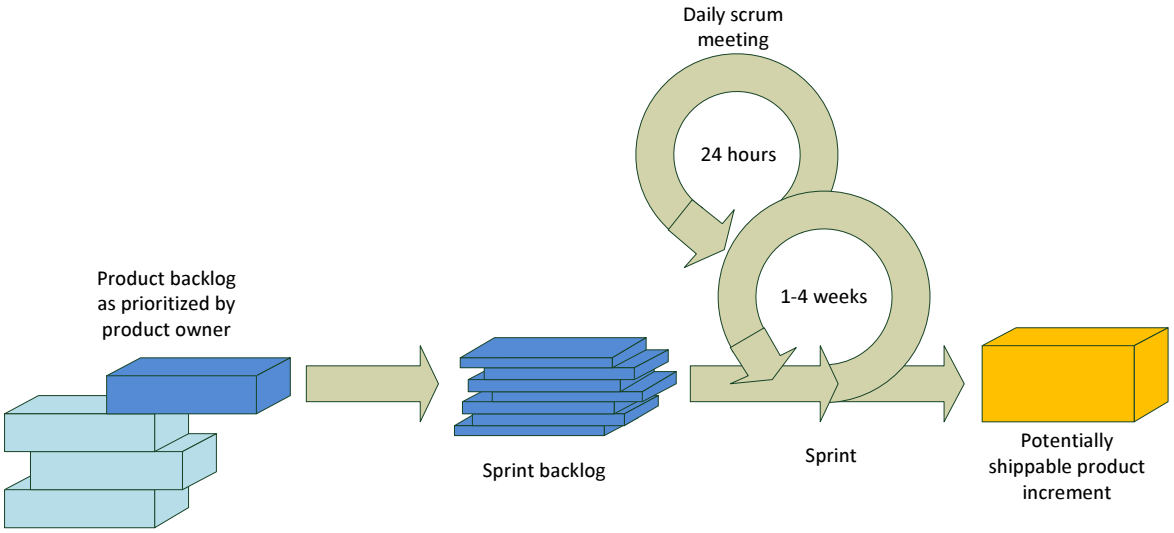


Figure 4. Scrum activities (Odne, 2010)

5.3.3 Agile Modeling

Many situations of software construction require developers to build a large and vital business function. The range and complexity of the software should be modeled, so that the software can be understood. Problems can be divided, becomes smaller, and quality of can be maintained at each step of software development. AM is a practical methodology that helps developers to make better documentation and modeling the software system. AM is a group of values, principles and practices for modeling software that can be applied on software development projects effectively. The principles of AM are 1) create a model in order to determine the purpose before making the model, 2) using multiple models: each model represents a different aspect of another model, 3) travel light: save the models that have a long-term course, 3) content is more important than appearance: modeling presents information to the right audience, 4) understand the models and tools that used to create software, 5) adaptation locally (Ambler, 2002).

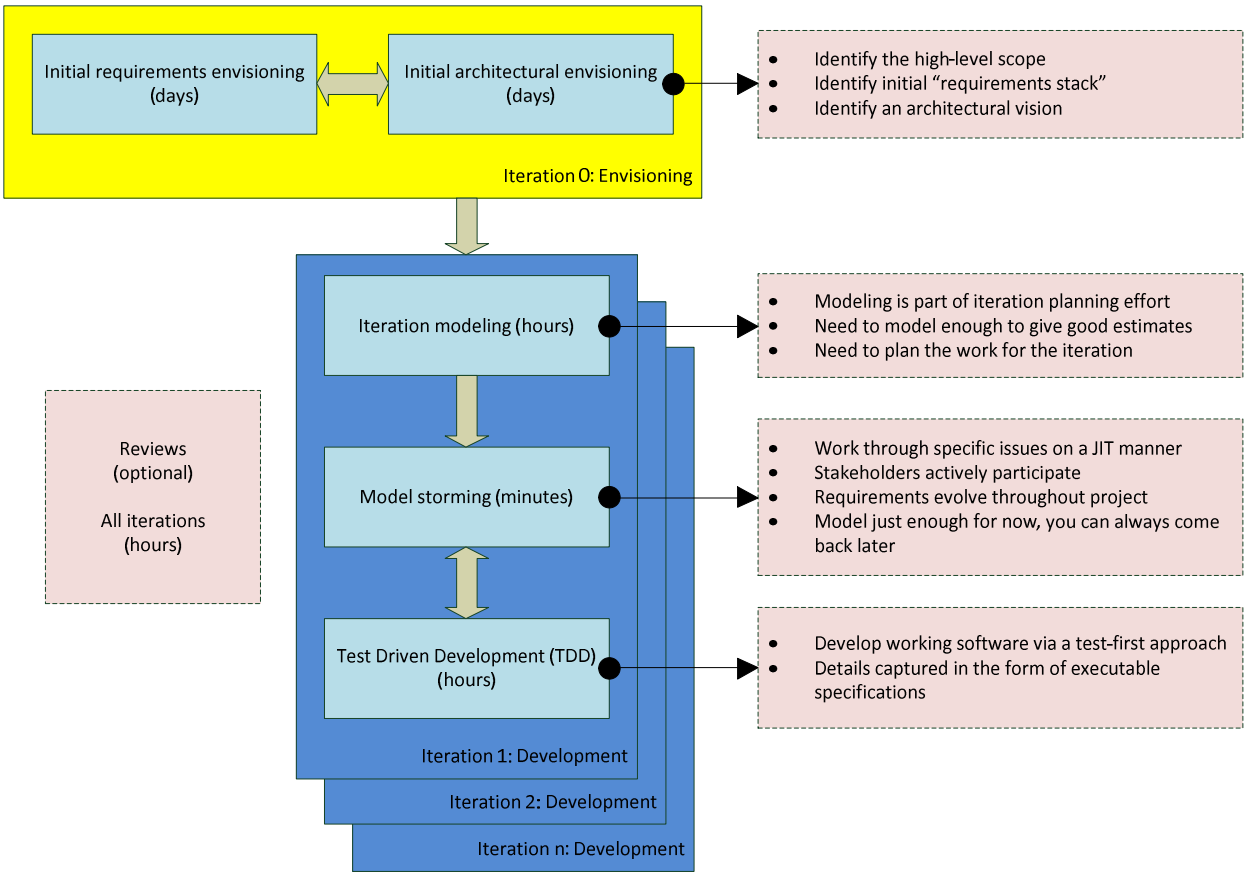


Figure 5. Agile Model Driven Development (AMDD) (Ambler, 2002)

5.3.4 Adaptive Software Development

ASD has been proposed by Jim Highsmith as a technique for building complex software and systems. The underlying philosophy is human collaboration and team self-regulation. Activities that occur during ASD process model are Speculation (Planning), Collaboration and Learning. Speculation (Planning) activity: adaptive planning cycle that uses initial information such as "missions" from client, project constraints and basic needs to be defined in series of software increments (software product, which is periodically submitted). Collaboration activity: people who are highly motivated to work together, complementary, willing to help, work hard, skilled in their fields, and communicate problems to produce an effective solution. Learning activity: development team often thought they already knew everything about the project, but they are not always so. Therefore, this process makes the team learn more about the project through three ways: 1) focus group: the clients and users give input to the software development, 2) formal technique reviews: comprehensive ASD team performs review, 3) postmortems: ASD team performs introspection on the performance and processes (Proboyekti, 2008).

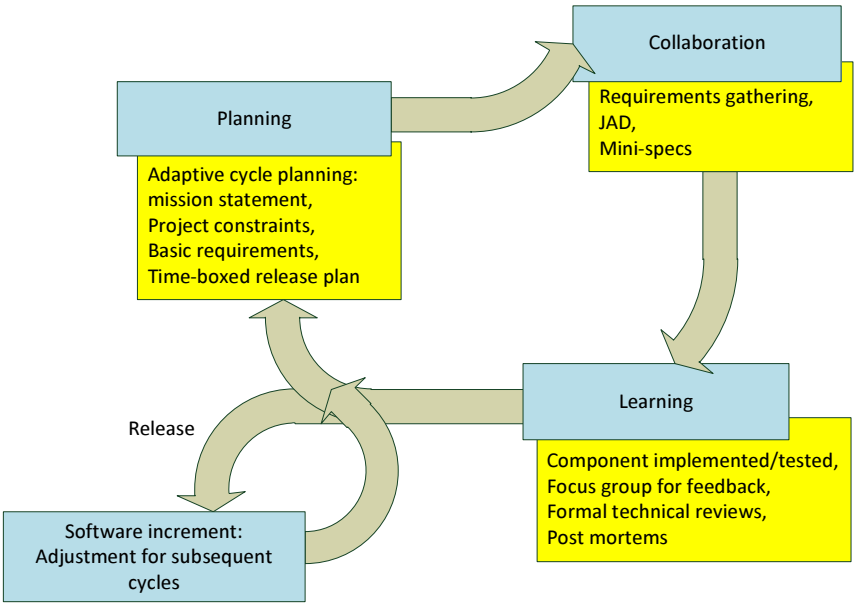


Figure 6. Workflow activity of ASD (Proboyekti, 2008)

6. FINDINGS

6.1. Literature Review

6.1.1. Review Methods

For this literature review, studies were eligible to put in the review if they presented empirical data on agile software development. Case studies of academic research that employed students of ICT program, and practical research that employed professional developers were included. Studies that focused on single techniques or practices, such as pair programming, scrum, xp, and collaboration-software-process were included. The studies were excluded if they did not present empirical data or if the main focus of the studies was not agile software development or otherwise was outside the scope of my study. Inclusion of the studies was not restricted to any year and specific type of articles, also the studies were written in English and Indonesian language. The review included qualitative and quantitative studies.

The reviewed studies met criteria for analysis by containing some theoretical evidence regarding customer & developer satisfaction, productivity, development costs, quality of produced software when use agile software development. In addition, the reviewed studies were containing some evidence regarding the effects of cultural dimension on adoption of agile software-development. Search strategy included electronic databases of articles such as conference proceedings, journals, books, etc. The following electronic databases were searched: ACM Digital Library, IEEE Xplore, Springerlink, and Google Scholar.

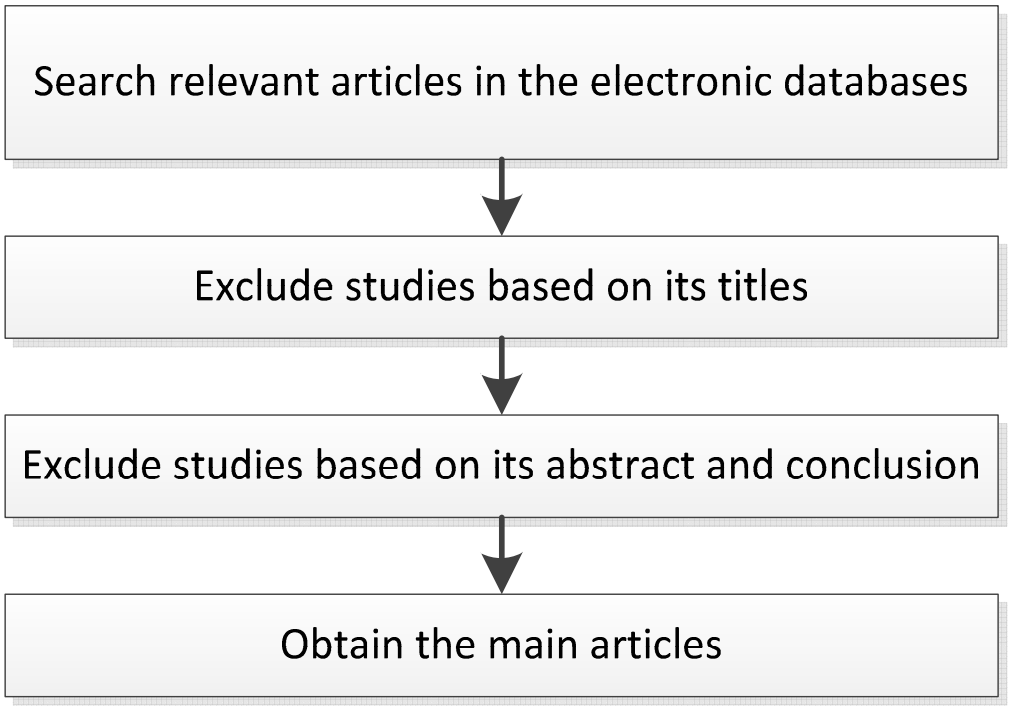


Figure 7. Phases of literature selection process

As shown in the Figure 7, systematic review process has four phases. Phase 1, search relevant titles, abstract, and keywords of the articles in the electronic databases using the following search key terms: agile software development, agile methods, agile vs. waterfall, agile adoption, benefits of agile methods, the success and failure of agile methods, scrum, extreme programming, xp, collaborative programming. Relevant citations were entered into Excel.

Phase 2, exclude the studies from its titles. Determine the titles of all studies that resulted from phase 1, whether they are relevant to the systematic review. Since I got several hits on the articles related to agile. At this phase, I excluded the studies that were clearly not regarding agile software development. Or, the studies were not empirical and were not independent. However, the titles were not always a clear description of what an article is about.

Phase 3, exclude the studies on behalf of its abstracts. At this phase, I excluded the studies that were not focused on agile software development. Or, the studies did not present empirical data. However, sometimes abstracts were giving poor description, misleading, and little indication of what was in the full article. The abstract was not always clear whether a study was empirical or not. Therefore, I included all studies that indicated a form of experience of implement agile software development. They consisted of information that met the criteria for analysis.

Phase 4, obtain the primary articles and studies. At this phase, I have had some studies that I thought were relevant to my research study, and would satisfy the research questions. In each article, there is a bibliography that presented some titles of other studies that would have relevance for my research study. Therefore, I picked some titles then reenact the systematic review phases, and so on.

6.1.2. Data Extraction and Synthesis of Findings

During phase 4, data was extracted from each of all main studies included in this systematic review according to a predefined extraction form (Tore Dyba, 2008).

Table 1. Data extraction form

Study Identifier	=>	Unique id for each study
Bibliographic reference	=>	Title, author, year, source
Study aims and objectives	=>	The aims and objectives of the study
Sample description	=>	Size, professional, students
Study's setting	=>	Industry, products, practices and processes used
Control group	=>	Yes/no, number of groups, sample size
Data collection	=>	Interviews, questionnaires, forms
Data analysis	=>	Qualitative, quantitative
Findings and conclusions	=>	The findings and conclusions from the study

This form helped me to record full details of the articles under review and to be specific regarding relevance of each article with my research questions.

In order to synthesize the extracted data from the main studies, I employed seven phases of meta-ethnographic methods (George W. Noblit, 1988), as presented below:

1. Getting started
2. Deciding what is relevant to initial interest.
3. Reading the studies
4. Determining how the studies are related.
5. Translating the studies into one another
6. Synthesizing translations
7. Expressing the synthesis

In a meta-ethnographic synthesis, studies can relate to one another in one of three ways: they may be directly comparable as reciprocal translations; they may stand in opposition to one another as reputational translations; or taken together they may represent a line of argument (Britten N., 2002).

This process of reciprocal and reputational translation and synthesis of studies achieved three things with respect to answering our overarching question about the benefits and limitations of agile software development (Tore Dyba, 2008). First, it identified a set of higher-order interpretations, or themes, which recurred across studies. Second, it documented that agile software development contains positive and negative dimensions. Finally, it highlighted gaps in the evidence about the applicability of agile methods to software development.

6.1.3. Results

I identified 33 empirical studies on agile software development. I split the studies into two groups. The first group, containing 30 studies, included survey, report, interviews, questionnaires, and case study investigating impacts and effects after introducing of agile software development. The second group, containing three studies, reviewed the adoption of agile development methods.

With respect to the kinds of agile method that have been studied, as shown in the Table 2, nine studies (30%) reviewed agile methods in general, six studies (20%) reviewed Extreme Programming practice. Five studies (17%)reviewed other methods that refer to internal methods, which used in the study. Scrum and Pair-Programming have the same portion in this studies overview, which are four studies (13%). Two studies (7%) reviewed Collaborative Software Process.

Table 2. Agile methods used in the study

Agile method	Amount	Percentage
General	9	30
Extreme Programming	6	20
Other	5	17
Scrum	4	13
Pair Programming	4	13
Collaborative Software Process	2	7

As mentioned above, the first group of studies contains surveys, reports, interviews, questionnaires, and case study reporting the effects and impacts after adopting agile software development. The following aims from each study are described in the Table 4.

Table 3. Study aims on introducing agile software development

Study	Study aim
S1	measures the interest in agile methods
S2	gains insight into the status of organizations currently implementing agile
S3	tests a research model that hypothesizes the effects of five characteristics of agile on stakeholder satisfaction
S4	develops an agile model for managing overtime, performance, and cost
S5	examines effects of Scrum implementation in a software company, in terms of overtime and customer satisfaction
S6, S26, S27, S28, S29	reports experience implementing agile methods
S7	examines whether the use and result of agile methods are as effective as the use and result of plan-driven methods in terms of customer satisfaction
S8	examines the effectiveness of pair programming on economics, satisfaction, design quality, problem solving, team building and communication, and staff and project management
S9	compares Personal-Software-Process and Collaboration-Software-Process effects on the productivity, cycle time, and product quality.
S10	presents a real case of agile customer engagement showing prerequisites, benefits, costs and risks in a software product setting
S11	compares QA techniques between Extreme Programming and Spiral Model
S12	compares QA abilities and frequency between agile practice and waterfall model
S13	investigates whether agile methods change and improve project management practices in software companies
S14	analyzes the effects of using XP for constructing commercial software
S15	compares XP and traditional-approach's performance
S16, S17	measures the success rate in the use of agile methods
S18	observes the effects of transition from poor and individualized programming approach, to extreme programming practice
S19	compares the quality and productivity when using pair programming and solo programming
S20	develops a data and workflow management system for scientists conducting clinical research
S21	compares the state of the art investigating issues and advantages when using agile and incremental development models
S22	investigates the perception of the bottlenecks, rework changes, and avoidable work, when migrating from a plan-driven software development to agile practice
S23	compares the programming performance when using pair programming and individual programming
S24	observes the impacts of collaborative software development to the software engineering course
S25	investigates the impact of pair programming on student performance, persistence, and

	perception
S30	analyzes evolution patterns for a system developed using XP

I found twenty-nine studies suggesting that implementing agile software development benefits the adopter. Five reports regarding experience of agile adoption (S6, S26, S27, S28, S29), show us five different companies that were switching from traditional development methods to agile software development. The reports tell how agile could work in big-scale projects, and the agile team delivered high-quality software successfully even at the first time of agile adoption. Agile adoption gave actual benefits to the adopters such as time-to-market was decreased, team productivity was increased, and customer satisfaction was increased. This finding indicates that agile is not only for small-scale projects, but it works for big projects as well.

All this time, people think that big scale projects that consisting of more than 10 people in an agile team, is not ideal for agile, and it would become a great obstacle for agile projects itself. Study S6 reports about Lockheed Martin's experience regarding adopting agile with more than 200 co-located people (distributed locations). According to the reports, implementation of agile was difficult and requires the right tool for this type of agile project, but at last the agile project achieved success with a 10% increase in productivity, product quality, and customer satisfaction. This finding does not correspond with how people normally think about agile. Obviously, agile possible to be used for a big project with hundreds of people in it, even at distributed locations. In addition, as agile gives positive effects on the company projects, agile affects software-engineering students as well. This is stated in five studies (S9, S15, S23, S24, S25). In general, the five studies reveal that students using agile to complete assignments have higher productivity than students using traditional methods.

Ten comparative studies (S9, S11, S12, S13, S15, S19, S21, S23, S24, S25) that have compared agile software development and traditional methods, found some findings that showed superiority of agile software development over traditional methods. The findings told that agile has advantages as the claim that is investigated in this research. Indeed, agile leads to better performance and productivity of development team, and faster time-to-market. Agile provides higher software quality and customer satisfaction than traditional methods do. However, I found one study (S7) by Donald L. Buresh that according to him, both methods satisfy their respective customers under a wide range of different situations. According to him, in company reports or case studies of a company, agile and traditional methods cannot be compared with each other. Since, each method was implemented in different projects with different initial conditions, people, and circumstances.

The second group of studies (S31, S32, S33) reviews adoption of agile software-development methods, affected by national and organizational culture. There are three studies reviews it. The first one, reviews regarding how national culture can affect adoption of agile methods in Europe, USA, and Australia. The author concluded that the chances for successful adoption of agile methods are strongly related to a low masculinity index where women and men have the same modest, caring values. Another one is low acceptance of the power distance index in which the less powerful members of society accept and expect that power is distributed equally. The two remaining studies review organizational culture. In

summary, the organizational culture aspect that has most effects on the adoption of agile methods is the human aspect. The human aspect includes the mindset of people in an organization, how they are thinking and understanding the concept of agile software development, how agile works for them, how agile is different from the old method. Mainly, this aspect is very influential in the executive level in the organization. Since, at this level all decisions are made.

6.2. Expert Interview and Web Survey

I employed two ways to collect data from the field of work, personal interviews and a web survey. I have made two kinds of questionnaires, an expert interview questionnaire (it can be seen in appendix b) and a web survey questionnaire (it can be seen in appendix c). The aim of these questionnaires was to find practical evidence from field experience of practitioners. The purpose was to gather information from software developers and other practitioners in the field of software development. It was regarding their experiences and opinions during they worked in software development projects.

These two questionnaires, both the personal interview questionnaire and the web survey questionnaire, have been piloted by my colleagues to know whether these questionnaires would be understandable and answerable, and my project leader was assessing whether it was proper or not.

6.2.1. Expert Interview

In personal interviews, I interviewed three experts of agile software development, actually they are agile consultants. Two persons from the Netherlands and one person from Indonesia. Since I am not a trained interviewer, and I had a lack of language skills. It was rather difficult to collect data in the interview way. As expected earlier, sometimes interviewees did not understand what exactly I meant with my questions. It was resulting in the answers that did not satisfy the interview questions. Therefore, I had to repeat the questions while explaining the point of the questions. In addition, the resulting answers were rather subjective, which makes data interpretation was difficult to be colligated.

As a note, results of interviews are subjective opinions regarding agile software development, since the interviewees are agile consultants. The results are approximately as follows:

There are two strongest points of agile implementations, the first is a mandatory use of requirements, because a lack of determined requirements would lead to more cost of the overall project. The second is early defect detection by early testing. Early testing enables to tackle the biggest risks first. It would save cost. The later defects are found in the project would lead to exponentially higher cost to solve the defects.

Actually, in the project, there are many stakeholders that cannot be satisfied by a function of software. There are business owners, managers, employees, even end users, customers of a customer. They have different requirements and interests on the software. So it is become quite complex requirements and problems that have to be solved. Developer teams normally cannot perfectly understand what the stakeholders want, since the stakeholders themselves do not know exactly what they really want. In the agile way, the developer team and business people will have a formal workshop, an interview, or they just sit together a whole day to determine and prioritize functional design based on business values and daily business activities. Communication between them is emphasized on direct communication, in one

place. They define and refine details of the requirements together. Indirectly, the business people help the developer team to understand the requirements from the business people better. Understanding of the requirements makes it easier to interpret the all requirements into functions of software. It has to be done in full-day planning sessions to capture the details of the requirements. The first-half planning day, they determine the requirements and then decide what will have to do in the iteration. The second half planning day, they decide how to implement the requirements, and how it will look like. This kind of practice is performed in every iteration.

The sense of business is hard to understand by the developer team, including changes in business that influences software development. Therefore, agile emphasizes close communication between the developer team and the business people. In every increment, the stakeholder gets demo software. The demo software enables the stakeholder to monitor whether the changes in software development have been according to what they ask for.

There is a Scrum board or visual management that enables the stakeholders to know what the developer team is working on. In the end of every increment, the stakeholder gets demo software. They will know what they are getting. Then they can determine functional priority, what they want to develop or change. What they want to stop or drop. What new functions they want to put in for next iteration, since the stakeholders do not know what they really want and need at the beginning of the project.

Agreement between the developer team and the customer is made in the informal way, without rigid contract, and the agreement is done in every iteration. Sometimes the developer team cannot find technical solution for the asked requirements. Therefore, they will discuss other alternatives with the customer. It makes that both the developer and the customers are convenient.

The changes can be influenced by internal factors such as internal politics issues of the customer, external factors such as what competitors of the customer have in their products whereas the customer has not.

In years of experience in the agile practice, the agile consultants always get positives feedback from the project owners due to implementation of agile resulting in fully meeting their needs, better product quality, lower cost, and shorter project duration. These benefits may not exist if the project is done by using traditional methods. In order to examine the feedback from the project owners, the agile consultants employ a formal survey or interview, a satisfaction index, or just looking at the customer expression and asking about what they feel.

6.2.2 Web Survey

In another way, I employed a web survey to collect data from software-development practitioners. I planned to use this method towards the end of the project. I made the web survey questionnaire, gathered email address of prospective participants, and I was contacting the prospective participants in less than two months approaching deadline of my project. I have tried to made survey questions as simple as possible and as straightforward as possible to be understood and answered. I made the questions in multiple-choice. I hoped the answer options would ease the participants to answer, and help the participants to determine the answers faster. I expected the participants would think the web

survey would not take too much time, or it would not be wasting their time. However, behind the simplicity of the survey questions, I did not ignore its effectiveness to provide information that should be obtained to satisfy the research questions.

The web survey took two weeks. It was run from May 16th until June 1st. It was too short for a web survey. However, I needed to collect data in a bulk in the short time due to the deadline. By using this method, the questionnaire of the web survey was quite easy to be broadened via email, forum website, mailing-list, and other Internet media. Yet, it was not about just spread the questionnaire out. From hundreds email invitations, there are only dozens responses from people who got invited to fill the questionnaire in. Most people ignored the web survey, since they thought the web survey did not benefit them, also it was wasting their time. I sent email invitations to more than 279 people. I also announced the web survey in several software-engineering-forums websites in the Netherlands and Indonesia. Until the deadline of the web survey, I just got nine teen responses. It provided insufficient data, but at least I could utilize the data for my research.

Before I sent the email invitations to the prospective participants, I assumed that they were working in the field of software development. As I was assuming, 58% of respondents claimed to be developers, 11% claimed to be IT manager and QA/tester, and 5% claimed to be project manager. Average of the respondents, have 2-5 years of work experience (58%). The remaining was 21% of the respondents have 5-10 years of work experience. 16% of the respondents have 10-20 years of work experience. These findings indicate that the respondents were deserved to be resources for this research since they have a senior position in their organization. They have sufficient experience and knowledge in ICT field.

53% of the respondents reported that their organizations were working in (software) technology sector. 11% of the respondents were working in government. 5% of the respondents were working in the IT consultant sector. In average, their organizations were small organizations. It can be assumed by looking to how many people were employed in IT department within the organization. Smaller organization must need fewer IT staff supports. 42% of the organizations have less than 11 IT staff within their IT department. 21% of the organizations have 11-50 IT staff. The remaining (assumed as median-big organization) has 51-100, 101-500, and 501-1000 IT staff support within their IT department (each option has 10% of percentage).

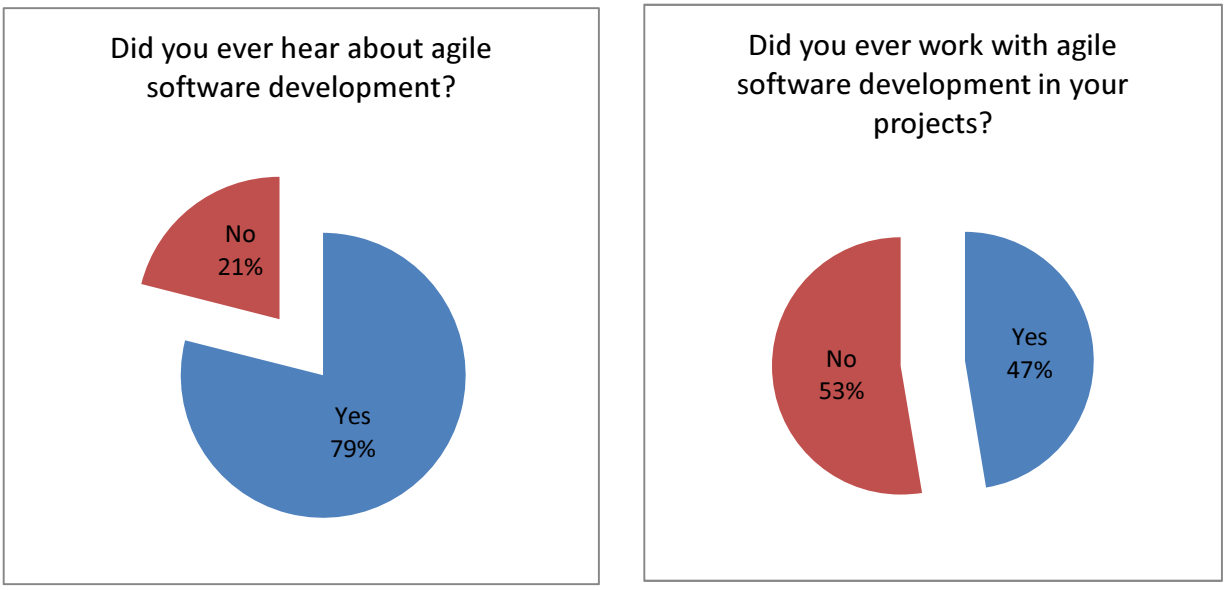


Figure 8. Appreciation of agile software development

About 79% of the respondents reported that at least they ever heard about agile software development. 53% of the respondents reported that they never worked in any agile development projects. The findings indicate that agile software development is quite popular as a software development approach. Even though, the adoption of agile software development was not too good, not as the popularity. 44% of the respondent said that lack of knowledge of agile software development including lack of knowledge of agile concepts and benefits was the biggest constraint in the adoption of agile software development in an organization. For people who do not know agile, it is impossible to propose agile methods in their software development projects. In addition, let us say the people know agile, but they do not know the concepts. These people will not take the risks to implement a method that is novel for them. The second reason why people do not apply agile is organizational culture issues, 33% of the respondents reported it. It cannot be denied that the decision-making is greatly influenced by organizational culture, mainly by executives within the organization. The rest claimed that the constraints of the adoption come from the customers who are not willing to involve too much in the development process (11%), you cannot say an approach as agile software development without customer involvement in the project. 11% of the respondents reported that agile did not fit for their projects. Perhaps they thought the scale of the project is too big for agile. They thought agile only fits for small projects.

Over the past few years, the respondents reported that they had applied agile few times in their projects. This finding can be assumed that some of the respondents believed that agile worked for their projects. They were confident with performance of agile software development. Therefore, they re-applied it in other projects. 70% of the respondent reported that their organization had run 1-5 agile projects. 20% of the respondents reported that their organization had run 6-10 projects. The rest (10%) reported that their organization had run 11-20 projects. 60% of the organizations have been using agile

methods for 1-2 years. 30% of the organizations have been using it for less than one year. The rest (10%) has been using it for 3-4 years.

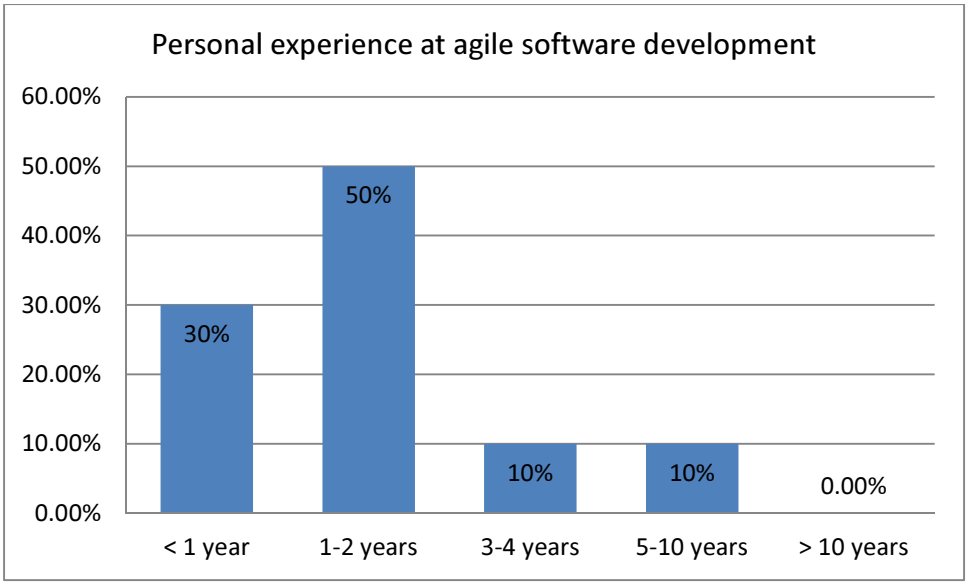


Figure 9. Personal experience at agile software development

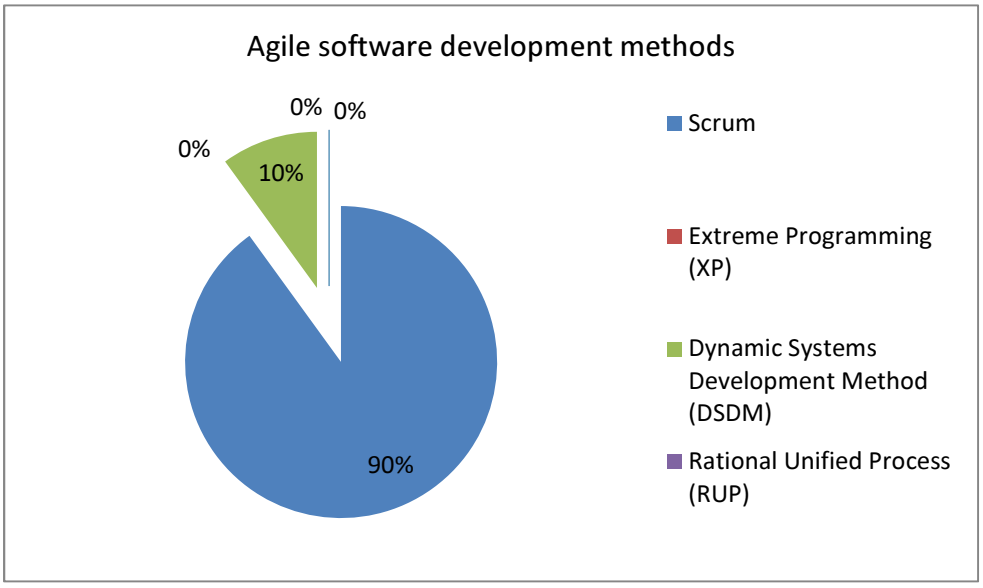


Figure 10. Agile software development methods

Personally, the majority of the respondents have sufficient experience with agile software development. In average, they have 1-2 years of experience with agile. It takes 50% of the respondents. More experienced respondents have 3-4 years and 5-10 years of experience, each in percentage of 10%. Additionally, 30% of the respondents have less than one year of experience with agile software development. From the survey's findings, the most popular agile method is Scrum. It takes 90%. The rest (10%) is DSDM (Dynamic Systems Development Method).

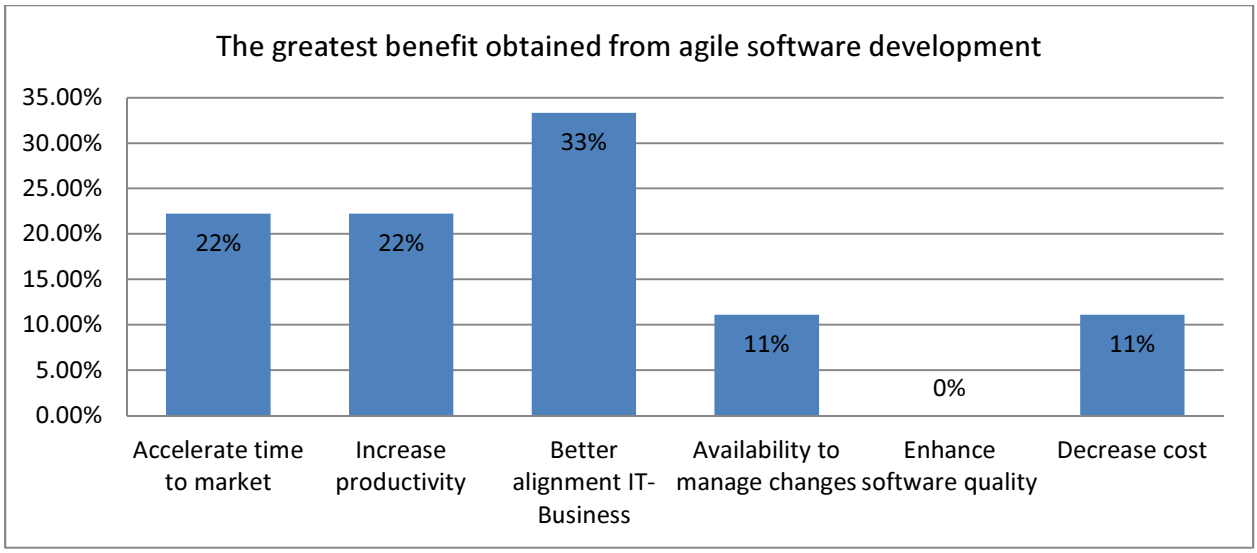


Figure 11. The greatest benefit obtained from agile software development.

The respondents as experienced agile users reported actual benefits of agile adoption that were: primarily better alignment IT-Business (33%), accelerate time to market (22%), increase productivity (22%), availability to manage changes (11%), and decrease development costs (11%). A surprising result showed that no one said that enhance software quality as the benefits of agile adoption. However, this survey has insufficient respondents. Probably, in case more respondents I got, the survey would have a different result.

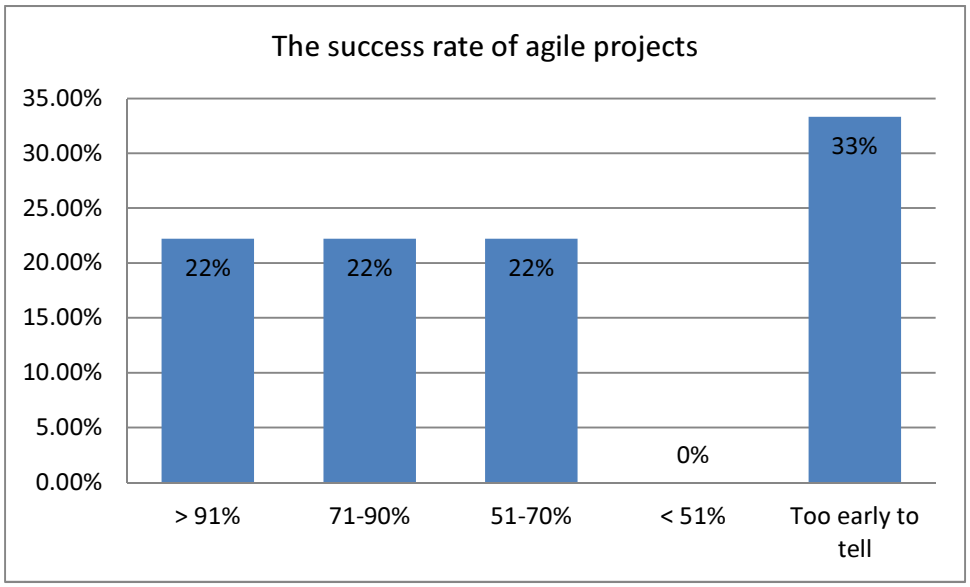
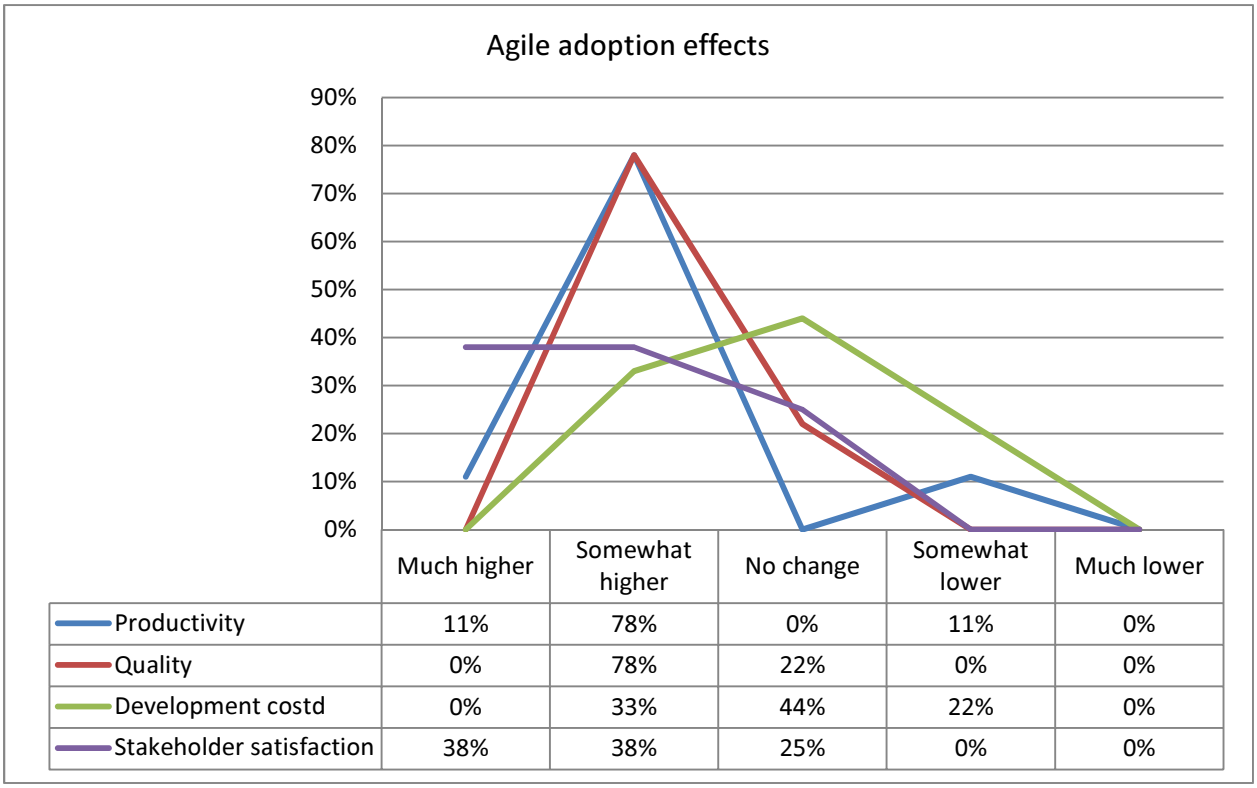


Figure 12. The success rate of agile projects

In terms of the success of agile projects, 44% of the respondents reported that more than 71% of agile projects were successful. 22% of the respondents reported that 51-70% of agile projects that they run were successful. The rest (33%) reported it is too early to tell.

In terms of the team size that has been successful with agile projects, 56% of the respondents reported that the largest team size they had been successful with agile projects was 6-10 people. “1-5 people” option was chosen by 33% of the respondents. 11% of the respondents choose “11-20 people” option. The findings did not meet my expectation. By this context question, I wanted to show that agile does not only fit for small sized team. I wanted to give a proof that agile fits for large sized team. 1-10 people in a team is the ideal number of people for an agile team. In case there were more respondents fill in the questionnaire, the result will be different.

Two questions addressed co-located agile teams. These two questions have been answered, 56% of the respondents reported that they had ever been involved in co-located agile teams, the rest (44%) said no. Co-location means that not everyone is on the same site during the development process. It can be different location, building, city, or even country. From the 56%, 50% of it cannot say the success rate of co-located agile teams. 33% of the 56% said that co-located agile team has 51-70% success rate. The rest (17%) said that co-located agile team has more than 91% success rate. By this context question, I wanted to prove that agile will works for co-located team as well as in the on-one-site team.



	Much higher	Somewhat higher	No change	Somewhat lower	Much lower
Productivity	11%	78%	0%	11%	0%
Quality	0%	78%	22%	0%	0%
Development costd	0%	33%	44%	22%	0%
Stakeholder satisfaction	38%	38%	25%	0%	0%

Figure 13. Agile adoption effects

In terms of the effects of agile adoption on performance of the team and resulting software, 78% of the respondents who experienced as agile users stated that agile led to higher productivity. 11% of the respondents reported that agile led to much higher productivity. The findings are in accordance with some of the findings of the studies that have been discussed in the literature review that agile adoption leads to higher productivity. Otherwise, 11% of the respondents said that agile adoption led to lower

productivity. The adoption of agile led to higher quality was reported by a 78% of respondents. These findings are in accordance with the literature's findings that have been discussed in the literature review. 22% of the respondents said that there was no change on the quality of the producing software. 44% of the respondents reported that there was no change between before and after used agile. 33% of the respondents realized that agile adoption led to a higher development cost, a 22% realized that agile adoption have led to a lower development cost. 67% of the respondents who experience with agile reported that their stakeholder was involved in their agile projects. These findings are in accordance to the agile principles which the business people and the developers work together throughout the project. The support of stakeholders and users is required to maintain a sustainable development. The stakeholder involvement helps the developer to understand the business process, it eases the developer to interpret the business needs and the business requirements into the functions of software that can help improve the business process. 38% of respondents reported that their stakeholder felt more satisfied after agile adoption, a 38% reported that the stakeholder much more satisfied, a 25% reported that there was no effect on stakeholder satisfaction.

In open questions, some of the respondents gave their opinions about the three most important things that make agile successful. In general, they reported: 1) members of the development team have to know their roles and responsibilities in the team, 2) everybody has to know how to apply the method, 3) close the gap between the customer and the developer by getting important stakeholders involved during the development process. Regarding the three most important problems that encountered with agile software-development methods, some of the respondents stated: 1) having the right stakeholders available at the right time, 2) hard to estimate the time, 3) unbalanced team members in terms of discipline and experience.

7. CONCLUSIONS & RECOMMENDATIONS

From the agile software-development discussion with the agile consultants, I got three important points:

1. Communication has an important role in the software development,
2. Software requirements are not easy to be identified completely at the beginning and volatile,
3. Co-operation & working together in the team (clients, users, and developers) determine the fluency of software development.

These three points are less accommodated by the traditional approach in the software development process. Since, traditional approach is designed to develop software in the rigid way. Everything must be according to plan. Everything must be according to signed contract. There is no flexibility, there is no tolerance, there are no changes after the contract has been signed. The customers will only get the resulting products according to the contract or even less. However, the customer needs will always change over the development time. The resulting products that cannot accommodate the needs of the customer definitely impact to acceptances and satisfaction of the customers.

Superior of agile software development over traditional methods is flexibility and tolerance of agile to changes, such as the changing requirements of software, the changing team, the changing technology, etc. Agile software development accepts the changes towards the end of software development in order to meet the customer's desires and needs. Agile has principles to provide a product that actually can be useful for the users and can help the business process of the clients. Agile also offers faster development time with minimal defects, high acceptances and satisfaction of the customers. How is it obtained? It is by Early coding, early testing, early customer feedback, and early bug fixing. These all activities are done repeatedly. Different from traditional methods that only have customer feedback at the beginning stage of development, and only have testing & bug fixing at the end stage of development.

In relation to the research questions, in the literature review, I have found twenty nine studies. They provided evidence that agile software development led to lower development costs, better software quality, higher productivity and customer satisfaction. According to S31, the chances for successful adoption on agile methods were strongly related to a low masculinity index, and a low acceptance of the power distance index in the national culture context. In the organizational culture context, the adoption of agile was strongly related to human factor in the organization. From the findings of the interviews and the web survey, I got evidence from the working field that agile methods were really used in the software development projects, and it successfully worked. I have found the proofs that agile concepts and processes can be understood well by the team and achieved success even in the first adoption.

Evaluation

Previously, my research project was titled “Agile Software Development and Business-IT Alignment." The purpose and the objectives of this project were similar with my current project. The difference was I was expected to make interviews with some business people as my research resources from the business point of view. I had to dig up information relating to opinions of business people after they used or implemented agile software development as a solution for their software development projects. The business people here can be stakeholders, clients, or end users who are utilizing the resulting software of agile software development. I had to gather information from them: what they know about the agile concept, what they are thinking about the customer involvement concept, what they did during software development, what they got and learnt from the agile software development process. It was quite difficult to find business people who are willing to spend their time to be interviewed as research resources. The same with finding literature that reviews topics related to customer satisfaction or agile software development from the business point of view, it was rather difficult since almost all the existing literature reviews agile software development from technical point of view. My project leader was also considering this research was a difficult project to find enough resources in the short time. Therefore, the subject of my research was changed to “The Success of Agile Software Development”. It was changed in the middle of the project time.

At that time, I had finished the agile expert interviews and some part of the literature review. The obtained data could still be used in the research results. However, since the project was changed and due to the approaching deadline, I had to use a web survey as an obtaining-data method instead of conducting other interviews. When I was using a web survey, I could obtain more data in the short time than by using interviews. I have sent hundreds of email invitation to the prospective participants, and I also have announced the web survey in some IT-forum websites in order to get as much respondents as possible. However, due to the remaining time, I was just running the web survey in two weeks. It was resulting in insufficient data that I obtained. But at least I had data to be processed as research results. Perhaps, in case I would have had more time, I could get more respondents, more data, and different results.

Subjective data such as the interview results and the answers of the last two web survey questions, it was rather difficult to colligate all subjective opinions from the research resources and then represent it in the report. In terms of conducting interviews, it was rather difficult as well since I have a lack of language skills. Therefore, I had to prepare the interview questions that it might provide the needed information. The questions of both the interviews and the web survey have been piloted by my colleagues to know whether they would be understandable and answerable, and my project leader was assessing whether it was proper or not. In addition, I used a voice recorder during the interview instead of writing the minutes for the lack of language skills reason.

Bibliography

Agile Alliance. (2012). Retrieved March 2012, from Agile Alliance: http://www.agilealliance.org

Ambler, S. (2002). *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process.* New York: John Wiley & Sons, Inc.

Beijleveld, M. (2011, April 24). *Cultural Dimensions and Agile Adoption*. Retrieved March 19, 2012, from ABC-thinkBIG: http://weblog.abc-thinkbig.com/#post100

Britten N., C. R. (2002). Using meta ethnography to synthesise qualitative research: a worked example. *Journal of Health Services Research and Policy*, 209–215.

Buresh, D. L. (2008). *Customer Satisfaction and Agile Methods.* IEEE Reliability Society.

Cemuturi, M. (2011). *Measuring Customer Satisfaction Using Internal Data.* Ezine Articles.

Chris, M. a. (2005). A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. *ADC '05 Proceeding of the Agile Development Conference*, (pp. 70-79).

Cockburn, A. (2002). Agile Software Development: The People Factor.

Corporate Report. (2003). *Agile Methodologies Survey Results.* Victoria, Australia: Shine Technologies Pty Ltd.

CrossTalk. (2002). Agile Software Development. *The Journal of defense Software Engineering*.

Dan Turk, R. F. (2002). Limitations of Agile Software Process. *Proceedings of the Conference on Extreme Programming and Agile Processes in Software Engineering.*

Dean Leffingwell, D. W. (2003). *Managing Software Requirements: A Use Case Approach.* Boston: Pearson Education, Inc.

Diane E Strode, S. L. (2009). The Impact of Organizational Culture on Agile Method Use. *Proceedings of the 42nd Hawaii International Conference on System Sciences* (pp. 1-9). Washington DC, USA: IEEE Computer Society.

Diane E. Strode, S. L. (2009). The Impact of Organizational Culture on Agile Method Use. *Proceedings of the 42nd Hawaii International Conference on System Sciences* (pp. 1-9). Washington DC, USA: IEEE Computer Society.

Dingsoyr, T. D. (2007). *Applying Systematic Reviews to Diverse Study Types: an Experience Report.* Madrid, Spain: IEEE Computer Society.

Dingsoyr, T. D. (2008). Empirical studies of agile software development: A systematic review. *Journal Information and Software Technology*, Pages 833-859.

Ferreira C., C. J. (2008). Agile System development and Stakeholder Satisfaction: A South African Empirical Study. *Proceeding of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding The Wave of Technology* (pp. 48-55). Wilderness, South Africa: ACM.

Ferreira, C. a. (2008). Agile System development and Stakeholder Satisfaction: A South African Empirical Study. *Proceeding of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding The Wave of Technology* (pp. 48-55). Wilderness, South Africa: ACM.

George W. Noblit, R. D. (1988). *Meta-Ethnography: Synthesizing Qualitative Studies .* London: Sage Publications.

GH, H. (1984). *Culture's Consequences: International Differences in Work-Related Values.* Newburry Park, CA: Sage Publications.

Highsmith, J. a. (2001). *Agile Software Development: The Business of Innovation.*

Hugh Beyer, K. H. (2004). An Agile User-Centered Method: Rapid Contextual Design. *Extreme Programming and Agile Methods XPAgile Universe 2004* (pp. 527-554). Springer.

Ibrahim, N. (2007). An Overview of Agile Software Development Methodology and Its Relevance to Software Engineering. *Jurnal Sistem Informasi Vol. 2 No. 1*, 69-80.

Ilincic, R. (2008). *Examining Agile Management Methods and Non-agile Management Methods in Global Software Development Projects.*

Jeffries, L. L. (2003). *Extreme Programming and Agile Software Development Methodologies.* CRC Press LLC.

Kai Petersen, c. W. (2009). A Comparison of Issues and Advantages in Agile and Incremental Development. *The Journal of Systems and Software*, 1479-1490.

Kai Petersen, C. W. (2009). A Comparison of Issues and Advantages in Agile and Incremental Development. *The Journal of Systems and Software*, 1479-1490.

Khurana, H. (2011). Implementation of New Management Agile Technique for Reducing Overtime and Increasing Customer Satisfaction. *International Journal of Engineering Science and Technology*, 238-241.

Kniberg, H. (2007). *Scrum and XP from the Trenches.* Stockholm: Crisp.

McCauley, R. (2001). *Agile Development Methods Poised to Upset Status Quo.* SIGCSE Bulletin.

McConnell, S. (1996). *Rapid Development: Taming Wild Software Schedules.* Microsoft press.

McCracken D. D., M. A. (1982). *Lifecycle Concept Considered Harmful.*

McCracken, D. D. (1982). *Lifecycle Concept Considered Harmful.*

Melonfire. (2006, September 22). *Understanding the pros and cons of the Waterfall Model of software development.* Retrieved March 26, 2012, from TechRepublic: http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/6118423

Miller D., L. J. (2001). The people make the process: commitment to employees, decision making, and performance. *Journal of Management*.

Miller, D. a. (2001). The people make the process: commitment to employees, decision making, and performance. *Journal of Management*.

Ming Huo, J. V. (2004). Software Quality and Agile Methods. *the 28th Annual International Computer Software and Applications Conference.* IEEE.

Pekka Abrahamsson, O. S. (2002). *Agile Software Development Methods: Review and Analysis.* VTT Electronics.

Pressman, R. (1997). *Software Engineering: A Practitioner's Approach, 4th Edition.* McGraw-Hill.

Quinn R. E., J. R. (1983). A Spatial Model of Effectiveness Criteria: Towards a Competing Values Approach to Organizational Analysis. *Management Science*, 363-377.

Quinn, R. E. (1983). A Spatial Model of Effectiveness Criteria: Towards a Competing Values Approach to Organizational Analysis. *Management Science*, 363-377.

Rashina Hoda, J. N. (2009). *Don't Mention the 'A' Word: Agile Undercover.*

Rashina Hoda, J. N. (2011). *Supporting Self-Organizing Agile Teams: What's Senior Management Got To Do With It?* Wellington, New Zealand.

Rashina Hoda, J. N. (2011). *Supporting Self-Organizing Agile Teams: What's Senior Management Got To Do With It?* Wellington, New Zealand.

Reich, B. H. (2000). Factors that influence the social dimension of alignment between business and information technology objectives. *MIS Quarterly, Vol.24, No.1*.

Reich, B. H. (2000). Factors that influence the social dimension of alignment between business and information technology objectives. *MIS Quarterly, Vol.24, No.1*.

Shaw C., I. J. (2002). *Building great customer experiences.* New York: Palgrave Macmillan.

Shaw, C. a. (2002). *Building great customer experiences.* New York: Palgrave Macmillan.

Taylor P.S., G. D. (2006). Do Agile GSD Experience Reports Help the Practitioner? *International Conference on Software Engineering, Proceedings of the 2006 international workshop on Global software development from the practitioner* (pp. 87 – 93). ACM.

Taylor, P. G. (2006). Do Agile GSD Experience Reports Help the Practitioner? *International Conference on Software Engineering, Proceedings of the 2006 international workshop on Global software development from the practitioner* (pp. 87 – 93). ACM.

Tore Dyba, T. D. (2007). *Applying Systematic Reviews to Diverse Study Types: an Experience Report.* Madrid, Spain: IEEE Computer Society.

Tore Dyba, T. D. (2008). Empirical studies of agile software development: A systematic review. *Journal Information and Software Technology*, Pages 833-859.

Vavra, T. G. (2002). *Customer satisfaction measurement simplified: a step-by-step guide for ISO 9001:2000 certification.* American Society for Quality.

VersionOne. (2007). *State of Agile Development.* VersionOne. http://www.versionone.com/state_of_agile_development_survey/

VersionOne. (2008). *State of Agile Development.* VersionOne. http://www.versionone.com/state_of_agile_development_survey/

VersionOne. (2009). *State of Agile Development.* VersionOne. http://www.versionone.com/state_of_agile_development_survey/

VersionOne. (2010). *State of Agile Development.* VersionOne. http://www.versionone.com/state_of_agile_development_survey/

VersionOne. (2011). *State of Agile Development.* VersionOne. http://www.versionone.com/state_of_agile_development_survey/

Wikipedia. (2012, February 28). *Agile Software Development*. Retrieved February 2012, from Wikipedia: http://en.wikipedia.org/wiki/Agile_software_development

Zwicker, M. (2007). War Stories - Fighter Jets and Agile Development at Lockheed Martin. *Agile Journal A Techwell Community*.

Appendix A: Literature Summary

This part provides the summary of each study that was used in the literature study.

S1	Agile Methodologies Survey Results
Author	Shine Technologies Pty Ltd
Year	2003
Type	Survey

In 2003, Shine technologies run a web-based survey to measure the interest in agile methods. They received 131 responses, from different organizations ranging from an online computer library center to NASA. The results showed that agile has reduced cost (49%), better or significantly better productivity (93%), better or significantly better quality (88%) and better or significantly better business satisfaction (83%) (Corporate Report 2003).

S2	State of Agile Development
Author	VersionOne Inc.
Year	2007, 2008, 2009, 2010, 2011,
Type	Survey

Year	2007	2008	2009	2010	2011	x̄
Participants	1681	2319	2570	4770	6042	-
Respondents adopted Agile	73%	-	84%	90%	80%	-
Increased Quality	77%	68%	63%	65%	68%	68%
Accelerated Time to Market	83%	83%	63%	70%	71%	74%
Reduce Cost	66%	65%	75%	39%	49%	59%
Improved Alignment between IT & Business Objectives	73%	66%	65%	68%	68%	68%

Four obtained benefits were picked, those aspects highly directly impact to customer satisfaction, increased quality, accelerated time to market, reduce cost, and improved alignment between IT & Business objectives. In average, 68% respondents found that the product quality after implemented agile in the project is better than used other methods. 74% respondents stated that implemented agile accelerates time to market. 59% respondents found they paid less cost than used other methods. 68% respondents said that implemented agile improves alignment between IT & Business objectives.

S3	Agile System development and Stakeholder Satisfaction: A South African Empirical Study
Author	Carlos Ferreira and Jason Cohen
Year	2008
Source	SAICSIT '08 Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology, Pages 48-55
Type	Proceeding

In this study, the authors examined a research model regarding five characteristics of agile method's effects (iteration, test-driven design, feedback, continuous integration, and collective ownership) on:

1. Stakeholder satisfaction with the development process
2. Stakeholder satisfaction with the development outcome

They focused on the Scrum approach as the agile methods practice.

They found that all characteristics of agile methods lead to higher customer satisfaction. The results showed that the customer is more satisfied with the development process, also with the overall project outcome. More often customer feedback within development process, the customer helped the developer team to recognize necessary requirement's changes by allowing the customer to voice their desired changes. The customer was fully satisfied because they get what they wanted.

S4	Implementation of New Management Agile Technique for Reducing Overtime and Increasing Customer Satisfaction
Author	Harsimarjeet Khurana
Year	2011
Source	International Journal of Engineering Science and Technology, Pages 238-241
Type	Journal

The author developed a new agile management technique. It's named PEOR model (Performance Evaluation and Overtime Reduction). This technique is a development of Scrum, it will efficiently work on a small team. It has been developed for managing the work in the proper and systematic manner to avoid overtime, to monitor the performance of the employees regularly and the most important is cost.

The qualitative data is obtained from customer opinion discussion. The overall feedback from the customer was positive after the introduction of PEOR. The customers said that they would recommend using PEOR in the future. The customer was involved to see the development process. The customers appreciated the concept of twice daily meeting because everyone knows what is required from them, and the concept of twice daily meeting has led to less misdirected development and clearer understanding of both the requirement and the development process by both the customers and the developer team. The customers were very satisfied. The twice-daily meeting has helped the customers and development team to visualize the product on the daily basis rather than at the end of the product.

S5	A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction
Author	Chris Mann and Frank Maurer
Year	2005
Source	ADC '05 Proceedings of the Agile Development Conference, Pages 70 - 79
Type	Proceeding

The authors conducted a study examines effects of Scrum implementation in the Software Company Petrosleuth. The case study examined overtime and customer satisfaction. They carefully examined the involvement of people, breaking down people's education levels, experience levels, and experience with agile methods.

The customers were happier with the produced software after the introduction of Scrum. They said it provided better consistency, transparency, and coordination. The customer mentioned feels more involved especially with the daily scrum meetings helping to keep them up to date. The only complaints were that the Scrum process is too rigid, and sometimes it is confusing to understand what the developers are working on.

S6	War Stories - Fighter Jets and Agile Development at Lockheed Martin
Author	Mark Zwicker
Year	2007
Source	Retrieved March 20, 2012, from Agile Journal: http://www.agilejournal.com/articles/columns/case-studies/313-case-study-war-stories-fighter-jets-and-agile-development-at-lockheed-martin
Type	Web site

In this literature, the author reports the details Lockheed Martin's experience adopting Scrum with more than 200 co-located people, which means in distributed locations. This paper revealed the greatest motivation for switching from their waterfall design to Scrum was a lack of customer satisfaction. Lockheed expected by adopting scrum the customer satisfaction would increase, higher than the old waterfall model. The important first step to introduce a team to Scrum is ensure everyone has information and knowledge how the Scrum process works and how to deal with any managerial and cultural impediments.

Working in a co-located environment is fundamentally more difficult and requires the right tools to keep in touch. Therefore, Lockheed had decided on Agile Enterprise, it allowed all project team members, as well as the shareholders, to access the status information of the project from a web browser. From the result of polling, this report claims a 10% increase in Productivity, Product quality, and Customer satisfaction.

S7	Customer Satisfaction and Agile Methods
Author	Donald L. Buresh
Year	2008
Source	Publisher: VDM Verlag Dr. Mueller e.K.
Type	Book

The author examined whether the use and result of agile methods as effective as the use and result of plan-driven methods in terms of customer satisfaction. Buresh employed some variables in order to examine it. The following variables are: project management effectiveness, project team effectiveness, and product quality.

Obtained data does not support the claim that the use and results of agile methods provide higher customer satisfaction than the use and results of plan-driven methods. He argues that both methods satisfy their respective customers under a wide range of different situations, it is the responsibility of the signers of the Agile Manifesto and the agile community to verify the propositions of the manifesto regarding customer satisfaction by using statistical analysis instead of case studies.

S8	The costs and benefits of pair programming
Author	Alistair Cockburn and Laurie Williams
Year	2001
Source	Book, Extreme programming examined, Pages 223 - 243
Type	Book section

In this paper, the authors examined the effectiveness of pair programming. They have investigated eight paths of software engineering and organizational effectiveness: Economics, satisfaction, design quality, problem solving, team building and communication, and staff and project management.

The authors revealed that pair the powerful point of pair programming is combining pair relaying and brainstorming. It enables faster problem solving, and by ongoing pair relaying and brainstorming resulting in better design and shorter code length. Continuous review by two programmers work collaboratively on the same programming tasks, they caught many mistakes and defects as they were coding than in QA test, and the end result has statistically lower defects. Significantly, the people who work in pair programming learn more about the system and software development. Since the knowledge is being passed between programmers, constantly, from tool usage tips to overall programming skills. The people also learn to work and talk together more often, giving better team dynamics and information flow.

S9	The collaborative software process
Author	Laurie Williams and Robert R. Kessler
Year	2000
Source	Doctoral Dissertation, Utah University
Type	Paper

The authors run an experiment examined PSP (Personal Software Process) and CSP (Collaborative Software Process) effects on the productivity, cycle time, and quality. They found that two programmers that work collaboratively produce high quality product faster than individual programmer. Additionally, they revealed that programmers more enjoy the software development process using CSP. Since the collaboratively programmers always have assist if they are confused by unknowing something. And the collaborative work of side-by-side programmers enables effective and efficient defects detection and defects removal, allows them to spend more time doing design and less time doing debugging.

S10	Agile Customer Engagement: a Longitudinal Qualitative Case Study
Author	Geir Kjetil Hanssen and Tor Erlend Fægri
Year	2006
Source	ISESE '06 Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, Pages 164 - 173
Type	Proceeding

In this longitudinal case study, the authors have followed a small software product company that has turned from a waterfall-like process to evolutionary project management (Evo). The case company is CompNN. It is a medium-sized Norwegian software company that provides a packaged software product for marketing and customer surveys called ProdNN.

Their analysis identifies a number of prerequisites for succeeding with this approach. Proactive stakeholder management is the foundation. Their analysis shows that CompNN has achieved a number of benefits as a result of Evo and the introduction of the PMT. First, close customer cooperation has a highly motivating effect on the developers. Second, developer's confidence has increased as a result of continuous settlement of expectations in that stakeholders assist in the prioritization of goals. The direct cooperation with users is a positive experience for the developers as it increases the quality of the communication and leads to improved understanding of the real business problems. Third, Evo has increased the visibility of the process internally in the organization and externally among the stakeholders.

S11	Software Quality Assurance in XP and Spiral - A Comparative Study
Author	Sajid Ibrahim Hashmi and Jongmoon Baik
Year	2007
Source	ICCSA '07 Proceedings of the The 2007 International Conference Computational Science and its Applications, Pages 367-374
Type	Proceeding

Quality Assurance is important to the quality of the software product regardless of the development process we choose. In this comparative study, the authors compared Quality Assurance techniques between Extreme Programming and Spiral Model that was developed based on waterfall model. The findings of the case projects indicate that code was produced by XP had less fault rate in comparison with the second method.

XP has closer link with VBSE (Value Based Software Engineering), and key elements of VBSE are addressed by XP. BRA (Benefit Realization Analysis) is performed prior to starting iteration in the form of informal meetings where pros and cons of the project are discussed. These kinds of meetings are also helpful to identify the non-software initiatives which may cause the realization of potential project benefits along with elicitation and reconciliation of stakeholders' value based conditions. The next element of VBSE deals with risk analysis and management that pervade the entire system life cycle. The productionizing stage of XP recommends slowing down the development so that risks can be identified and mitigated. Earned value management in VBSE tracks whether project is meeting its original plan. In XP, it is ensured in number of ways. Above all, acceptance tests that were performed by on-site customer ensure that project never deviates from its proposed functionality, value measures are considered properly while product is being developed. XP has also got a capability to change as an opportunity, taking effect both from inside and outside. Changing requirements from customer can be the result of change in market trends, or introduction of new technology. Whereas XP itself is flexible enough to let its users use its core practices according to their requirements.

S12	Software Quality and Agile Methods
Author	Ming Huo, June Verner, Liming Zhu, and Muhammad Ali Babar
Year	2004
Source	COMPSAC '04 Proceedings of the 28th Annual International Computer Software and Applications Conference - Volume 01, Pages 520-525
Type	Proceeding

In this comparative study, the authors have analyzed agile practice's quality assurance abilities and their frequency compared to waterfall model quality assurance abilities.

In the agile way, there is a general practice which called On-site customer. The customer helps developers refine and correct the requirements. The customer supports the developer team throughout the development process. In the waterfall model, customers typically involved just in requirement definition, possibly system, and design. They are not involved and do not contribute as much as in agile development.

Acceptance testing is a dynamic quality assurance practice. It's carried out after all unit tests have passed. In agile methods, this practice occurs much earlier and more frequently than the waterfall model. Earlier acceptance testing, earlier customer feedback. The small release enables the developer team to get customer feedback as early as possible, which provides valuable information for development process.

Continuous integration means the developer team does not integrate the code once or twice. The main purpose is to catch enough bugs to be worth the cost. Continuous integration reduces the time to search bugs and enables detection of error early. The agile development integration is done much earlier, and its frequency is much higher than the waterfall model.
Pair programming, also known as continuous code inspection means two programmers work on the same code, continuously. It improves design quality and reduces defects by continual design and code review process.

S13	Project Management in Plan-Based and Agile Companies
Author	Martina Ceschi, Alberto Sillitti, and Giancarlo Succi
Year	2005
Source	Journal IEEE Software archive, Volume 22 Issue 3, May 2005, Pages 21 - 27
Type	Journal

The authors had conducted an empirical study to investigate whether agile methods change and improve project management practices in software companies. They analyzed that the main problem in software development is delivering products with all the features on time. Approximately 50 percent of the plan-based companies and a 10 % of agile companies believe they have a difficult relationship with their customers. However, agile companies' customer relationships are not so difficult to manage. In fact, one of the main problems that was solved by adopting agile methods is the customer relationship.

Both agile and plan-based companies have collaborative customer relationships. A 60% of the agile companies have their customers on-site. A 40% of plan-based firms use this practice as well. Normally, it is difficult to keep the contract fixed over project. Agile companies tend to deal with their costumer in flexible contract instead of the fixed ones. Constant customer involvement allows quick response to changes by the development team. The higher customer involvement increases quality link between the customer and the development team. Therefore, agile companies are more satisfied with their customer relationships than plan-based companies. Plan-based companies with traditional development process difficult to respond quickly to changes. Every changes and modifications risk the project plan and the process of organization.

S14	Exploring Extreme Programming in Context: An Industrial Case Study
Author	Lucas Layman, Laurie Williams, and Lynn Cunningham
Year	2004
Source	ADC '04 Proceedings of the Agile Development Conference, Pages 32 - 41
Type	Proceeding

This paper describes a longitudinal case study that analyzed the effects of using XP for constructing commercial software at Sabre Airline Solutions. This study compares two releases of the same product. One release accomplished using a traditional waterfall model, and the other, after running 2 years, using extreme programming methodologies. All team members reported that communication among the team increased when using extreme programming, which allowed problems to be solved more quickly, and all developers benefited from increased customer feedback.

In comparisons of the new release to the old release show that programmer productivity increased by 50%, pre-release quality improved by 65%, and post-release quality improved by 35%. In fact, the defects causing the unusable system were reduced to 0% using XP from 12% using the waterfall model. From these findings proof that, over time, implementing the extreme programming results in increased productivity of developers and improved quality of code.

S15	A Formal Experiment Comparing Extreme Programming with Traditional Software Construction
Author	Francisco Macias, Mike Holcombe, and Marian Gheorghe
Year	2003
Source	ENC '03 Proceedings of the 4th Mexican International Conference on Computer Science, Page 73
Type	Proceeding

The author was running an experiment conducted on computer science student that was aimed to assess XP and compare it with a traditional approach. In terms of quality and size, extreme programming teams produced similar quality and size of the end products to traditional-approach teams.

According to the activities and the time spent, extreme programming teams spent more time in testing, and spent less time in programming. In the extreme programming approach, teams working spent much less time in analysis and design. Oppositely, traditional-approach teams spent less time in testing, and spent more time in programming. Traditional-approach teams spent much more time in analysis and design. And, extreme programming is a process requiring low technology, it means that less expensive than traditional approaches that require expensive and more sophisticated technology.
The final results showed that the use of XP in a software construction process produced as good results as obtained from the use of traditional approach, in both external and internal quality. The authors were surprised that, even though extreme programming is a much newer concept than the traditional approach, it can provide as good result as traditional approach, and given that XP procedure absence of Design phase provides as good results as traditional approach that including Design phase.

S16	2007 IT Project Success Rate Survey
Author	Dr. Dobb's Journal (DDJ)
Year	2007
Type	Survey

In 2007, IT Project Success Rate Survey was run to measure the success rate in the use of agile methods. The survey received 586 responses, from different organizations ranging. The results showed that the respondents reported that agile software development project have a 71.5% success rate, offshored software development projects a 42.7% success rate, and traditional projects a 62.8% success rate. 61.3% of respondents believed that delivering when the system is ready to be shipped is more important than delivering on schedule. 79.6% of respondents believe that providing the best ROI is more important than delivering under budget. 87.3% of respondents believe that meeting actual needs of stakeholders is more important than building the system to specification. 87.3% of respondents believe that delivering high quality is more important than delivering on time and on budget. 75.8% of respondents believe that having a healthy workplace is more important than delivering on time and on budget.

S17	2010 IT Project Success Rates survey
Author	Dr. Dobb's Journal (DDJ)
Year	2010
Type	Survey

In 2010, Dr. Dobb's Journal conducted an IT Project Success Rate Survey to measure the success rate in the use of agile methods with 203 respondents. According to the results, Agile and Iterative project teams have statistically identical success rates around 60%, ad-hoc project teams (no defined process) and traditional project teams have lower success rates than agile/iterative project teams, each has around 44% and 43%. 54% of respondents prefer to deliver on time according to the schedule and 44% prefer to deliver when the system is ready to be shipped. 35% of respondents prefer to deliver within budget and 60% prefer to provide good return on investment (ROI). 14% of respondents prefer to build the system to specification and 85% prefer to meet the actual needs of stakeholders. 40% of respondents prefer to deliver on time and on budget and 57% prefer to deliver high-quality, easy-to-maintain systems

S18	Using Extreme Programming in a Maintenance Environment
Author	Charles Poole and Jan Willem Huisman
Year	2001
Source	Journal IEEE Software archive, Volume 18 Issue 6, November 2001, Pages 42 - 50
Type	Journal

In this study, the author observed the effects of transition from poor and individualized programming approach, to extreme programming practice in the Iona Technologies. By the end of 1997, Iona Technologies had developed system code (called Orbix) that already patched and re-patched hundreds of times. This practice resulting in increased code entropy and decreased program understandability. Iona Technologies had lack in proper documentation process, and documentation visibility was scarce in their old methodology. Each software engineer had no focus on process improvement, and engineers reported not feeling cohesive in their software teams.

The authors concluded that after introducing extreme programming practice, there were fewer issues in new release, code entropy was significantly reduced, code complexity was decreased, and there were no patch rejections during the last months of this study. The findings suggest that adopting XP approach improved the developer team's productivity and ability to deliver quality support. The productivity was improved by 67% over the old employed practice. Furthermore, improvement continued, the team size was reduced from 36 to 25 developers. One of the greatest benefits to the team is visibility improvement.

S19	When does a pair outperform two individuals?
Author	Kim Man Lui and Keith C. C. Chan
Year	2003
Source	XP'03 Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering, Pages 225-233
Type	Proceeding

The authors performed experiments with experienced software developers. They reports experimental measurements of quality and productivity when using pair programming. In this experimental study, paired-programmer and solo-programmer group were requested to complete algorithm-style aptitude tests in order to observe the capability of solving algorithms in pairs and in solo.

The results establish a statement that a pair outperforms individuals in working on computer algorithms in terms of quality and productivity. They concluded that pair programming achieves higher productivity when a pair writes a more challenging program that demands more time spent on design. The finding explains that it is effective to write a program in pair for rapid changing requirements because it demands that programmers concentrate on changing (or continuous) design.

S20	Staying Agile in Government Software Projects
Author	Barg Upender
Year	2005
Source	ADC '05 Proceedings of the Agile Development Conference, Pages 153 - 159
Type	Proceeding

This project has 3 phase project involving the development of a centralized database and web application for the National Institute of Health that would house clinical research studies on human subjects in order to find better ways to detect, diagnose, treat and prevent a variety of diseases. Since the studies involve humans, the highest ethical and safety standards must be followed. The project had also had several failed attempts in the past due to the inherent challenges of unifying the requirements and the organization. The traditional waterfall-based contracting model was not able to address the technical and organizational challenges.

Overall, the Scrum practice helped the team to become more efficient in delivering the software. Some of the processes were difficult to adapt to the certain project, but agile practices have unquestionably helped the team, the project managers, and the end user. The team also reported better communication, more time writing software and visibility into progress that allowed them more freedom to make decisions. The users were pleased with the quick turn-around and better control over the direction of the system.

S21	A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case
Author	Kai Petersen and Claes Wohlina
Year	2009
Source	Journal of Systems and Software archive, Volume 82 Issue 9, September, 2009, Pages 1479-1490
Type	Journal

In this literature, the authors compared the state of the art investigating issues and advantages when using agile and incremental development models with an industrial case study where agile as well as incremental practices are applied. The articles considered in the state of the art are based on empirical studies. Regarding the research questions and contributions they concluded that adopting agile and incremental practices in large-scale software development leads to benefits in one part of the process, while raising issues in another part of the process.

In summary, the main advantages that have been revealed by this literature from the case study are: the requirements are more easier to estimate since the scope could be reduced, the need of documentation was reduced since employed direct communication in teams, testing resources are used more efficiently, frequent deliveries allow early feedback, rework reduction, higher transparency and visibility, low requirements volatility, and reduction of waste requirements.

S22	The Effect of Moving from a Plan-Driven to an Incremental Software Development Approach with Agile Practices
Author	Kai Petersen and Claes Wohlin
Year	2010
Source	Empirical Software Engineering archive, Volume 15 Issue 6, Pages 654-693
Type	Journal

This paper aimed to investigate the perception of the bottlenecks, rework changes, and avoidable work, when migrating from a plan-driven software development to agile practice.
The qualitative data showed that constructing the product in incremental approach using LSV concept allows higher release frequency. Ericsson was able to deliver the functionality more frequently than using plan-driven approach, which would benefit the organization, since frequent releases lead to earlier return on investments (ROI). In plan-driven approach, a large up-front investment is required which starts paying off when the overall development has been completed. The qualitative data showed that there is a clear improvement in the waste reduction after introducing agile practice. And, the requirements were better descripted when using agile practice than plan-driven, indicated by the number of change requests have been reduced. The quantitative data showed improvement in early testing done before system testing (LSV), reflected in a reduced fault-slip-through in comparison to the plan-driven approach. Furthermore, the constantly rising maintenance effort decreased after introducing incremental and agile practice. The amount of documentation can be reduced because much of the documentation was related to hand-overs between phases, direct communication can replace parts of the documentation. Furthermore, in plan-driven development the knowledge of people is very specialized and they have a lack of confidence. This can be hindering in the beginning when moving from plan-driven to incremental and agile practices as having small teams requires very broad knowledge of the team members. However, at the same time face-to-face interaction helps team members to learn from each other and gain insight and understanding of the overall development process.

S23	Experimenting with Industry's "Pair Programming" Model in the Computer Science Classroom
Author	Laurie A. Williams and Robert R. Kessler
Year	2001
Source	Journal of Computer Science Education, Pages 1-20
Type	Journal

The authors were running experiment in the Summer and Fall semesters at the University of Utah. The Summer class used pair programming in the Collaborative Development of Active Server Pages, whereas the Fall class used pair programming in a senior software engineering course.

All findings suggest that the paired-programming students almost delivered their products on time. And, the paired-programming students performed much more consistently and resulting higher quality. 95% of the class felt more confident in their assignments because they pair programmed. On average, students that worked in pairs passed 15% more of the tutor's test cases. The quality difference between paired-programming and individual-programming was statistically significant. The students felt they were more productive when working collaboratively. Students were happier and less frustrated, and were more confident in their work. A 92% of the students said they felt more confident in their projects when working in pairs, 96% of the students said they felt more enjoyed the work when working in pairs. On an anonymous survey, 84% of the class felt enjoyed doing the assignments more when working with partner.

S24	How and Why Collaborative Software Development Impacts the Software Engineering Course
Author	Lucas Layman, Laurie Williams, Jason Osborne, Sarah Berenson, Kelli Slaten, and Mladen Vouk
Year	2005
Source	Proceedings Frontiers in Education 35th Annual Conference, Pages T4C9-T4C14
Type	Proceeding

In order to observed the impacts of collaborative software development to the software engineering course, the authors gave paired programming assignments during software engineering classes during Spring and Fall 2004 at North Carolina State University.

The findings suggest that there exist positive correlation between saving time in paired-programming and a tendency to procrastinate when students work individually. The students believed that they were more organized when working in pairs, and it was resulting in higher productivity. The students saw pair programming as beneficial in uncovering logic errors than in uncovering coding errors. This statement suggests that even the most confident students, who do not benefit from pair programming to find code errors, can still benefit by explaining their thoughts to a partner to uncover errors in their logic. Results suggest that the most confident students, perhaps the best programmers, felt that they could be held back by less suitable partners. Conversely, paired-programming seems to benefit students with lower programming self-confidence.

S25	The Impact of Pair Programming on Student Performance, Perception and Persistence
Author	Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald
Year	2003
Source	ICSE '03 Proceedings of the 25th International Conference on Software Engineering, Pages 602 - 607
Type	Proceeding

During 2000-2001 academic years, the authors were running an experimental study. The authors held four sections of the course were offered during the year: one in the Fall quarter, two in Winter, and one during the Spring. Students enrolled in the Fall and Winter sections were required to complete all assignments using pair programming, whereas students in the Spring section were required to complete programming assignments independently.

All findings suggest that paired students were significantly more likely to complete the course than were non-paired students, and therefore more likely to pass the course as well. Students who paired reported significantly higher confidence in their program solutions than students who worked individually. Paired students reported greater satisfaction than non-paired students. Paired students more enjoyed working on programming assignments more than non-pairing students.

S26	Primavera gets agile: a successful transition to agile development
Author	Bob Schatz and Ibrahim Abdelshall
Year	2005
Source	Journal IEEE Software archive, Volume 22 Issue 3, May 2005, Pages 36 - 42
Type	Journal

This is an experience report of Primavera. It has developed using the waterfall model in the past and has ended up working late nights and weekends to finish projects on time, then, it has decided to try using Scrum. With the new Scrum process the teams began working 40 hour work weeks in 30 day sprints.

This article has revealed Primavera's success using Scrum, resulting in decreased reported defects and faster time-to-market. The article also points out the benefits to the team members that no longer have to work at an unreasonable pace and not having any turnover for 10 months. Even with the many successes implementing Scrum there were still problems along the way. One problem was too much attention being put on adding features each sprint rather than making sure there were no bugs in existing features. Another problem was with the requirements of the project constantly changing each sprint it is difficult for the stakeholders to know how much work is left to be done before release time. This study does a good job pointing out several possible pitfalls from adopting Scrum as well as showing the benefits in a quantifiable manner.

S27	Establishing the Agile PMO: Managing variability across Projects and Portfolios
Author	Ash Tengshe and Scott Noble
Year	2007
Source	AGILE '07 Proceedings of the AGILE 2007, Pages 188-193
Type	Proceeding

This article is about experience report of Capital One Auto Finance switching from a traditional waterfall model to agile approach. Capital One Auto Finance's IT division had an increasing problem of delivering business value on time along with growing customer dissatisfaction, so the time was right to have some positive changes. These positive changes included the exploration and implementation of an agile development process using Scrum.

This article goes in depth how the top-down switch to Agile from waterfall was successfully completed. One of the main goals of becoming Agile was to improve time-to-market and customer satisfaction. After 40+ Agile Projects were completed they found that the time-to-market was 50% faster and the customer satisfaction on all of the 40+ projects was 100%. These outstanding results validated their initial belief that customer collaboration was the key and agility with Scrum creates positive results.

S28	Ongoing quality improvement, or: how we all learned to trust XP
Author	Mark Striebeck
Year	2005
Source	ADC '05 Proceedings of the Agile Development Conference, Pages 267 - 271
Type	Proceeding

This paper is about experience VA Software adopted extreme programming. VA Software realized that the traditional waterfall model would not work for its project. The product managers were continuously making changes about which features to include or upgrade from the legacy product, and the team needed a more agile process to react to the changing requirements. Two project managers researched extreme programming and decided to apply it to the product.

One point of note is that this paper suggests that the team adopted an iterative-test-last process instead of test-driven development as mandated by extreme programming, tests suites were created and run continuously, both on a CruiseControl server and a Tinderbox server. The team saw development time decrease, while code quality increased. The last reported release of the product came in under budget by three calendar days. Quality was measured in number of bugs found per release; since switching to extreme programming practices, the product has experienced an 80% reduction in the number of bugs per man-weeks of product development.

S29	Experiences with Extreme Programming in Telehealth: Developing and Implementing a Biosecurity Health Care Application
Author	Ann Fruhling, Kimberly Tyser, and Gert-Jan de Vreede
Year	2005
Source	HICSS '05 Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 6 - Volume 06, Page 151.2
Type	Proceeding

This is an experience report with Extreme Programming in Telehealth. It describes the effects of adopting extreme programming for a project called STATPack.

Overall, the study concludes that extreme programming is an effective methodology to develop health care applications. The rapid prototyping enabled IT developers and health care users to clarify system requirements, communicate openly, and quickly build rapport. Further, the research found that where the technology was new or foreign, extreme programming was flexible enough to support several iterations of technology and produce prototypes in a timely manner. Extreme programming seems to lower management overhead according to the study, heightens team productivity, and better satisfies customers while building trust between those customers and the developers.

S30	An Empirical Study of the Evolution of an Agile-developed Software System
Author	A. Capiluppi, J. Fernandez-Ramil, J. Higman, H. C. Sharp, and N. Smith
Year	2007
Source	ICSE '07 Proceedings of the 29th international conference on Software Engineering, Pages 511-518
Type	Proceeding

This study presents the first measurement-based study of the evolution of software developed using an agile approach in an industrial setting.

Smooth growth was seen in the evolution of this agile system. Growth rate measured in lines of code was higher than growth rate measured in files or directories. The final product's quality was enhanced using extreme programming methodologies. The clients were impressed with the agile approach because of the responsiveness of the team to their needs. The team managed to produce better quality software for the customer. Refactoring in order to maintain code simplicity directly resulted in higher quality code, and having an on-site customer allowed programmers to build a system that was constantly evolving toward the customer's idea of the final system, removing the possibilities of producing waste code that implemented a misunderstood requirement. The product in this paper was also a success commercially in that it survived well in the market for 6 years, continuing to use extreme programming for the entire time.

S31	Cultural Dimensions and Agile Adoption
Author	Mary Beijleveld
Year	2011
Source	Retrieved March 19, 2012, from ABC-thinkBIG: http://weblog.abc-thinkbig.com/#post100
Type	Web

The author conducted comparison study between Geert Hofstede's comprehensive study on how values in the workplace are influenced by country's culture compared to Dave Norton and Rob Thomsett's insights on agile adoption in the Netherlands and other countries, when they gave an agile workshop in the Netherlands. According to Geert Hofstede's (1984) study, there are five cultural dimensions on which each country can be compared, Hofstede indexed on a scale from 0 to 100.

Mary concluded that the chances for successful adoption on agile methods are strongly related to a low masculinity index and low acceptance of power distance index such as in the Netherlands and Scandinavia and a low uncertainty avoidance index, Netherlands and Scandinavia's index lower than world average. In Belgium, high power distance index, for instance, it's much more important to first gain executive support for agile practices. In the Netherlands, you have to prove that agile works and gives sustainability. Belgium's higher score on uncertainty avoidance suggests less acceptance of change. Belgian decision makers might have a higher need for clear measures, rules and more waterfall-like methods. Netherland and certainly the UK and USA will be more open to other solutions. Germany is in between. Based on cultural differences in Belgium and Germany, chances for agile project methods to be adopted are less than in the other six countries. Furthermore, the UK, the USA and Australia seem culturally less inclined to adopt agile practices.

S32	Supporting Self-Organizing Agile Teams: What's Senior Management Got To Do With It?
Author	Rashina Hoda, James Noble, and Stuart Marshall
Year	2011
Source	12th International Conference on XP, Pages 73-87
Type	Journal

In this paper, the authors highlight the influence of organizational culture on the use of agile methods based on theory study of 58 agile practitioners across 23 different software organization in New Zealand and India. According to the author's knowledge that an environment of isolation, timidity, and secrecy will cause challenges. Their research supports the claim that an environment of openness, communication, and trust is imperative for self-organizing agile teams. They found that senior management support is critical environmental factor influencing self-organizing agile teams. The influence of senior management in creating and maintaining such an environment is extremely important. Since senior management influences the organizational structure and culture in an organization. Agile methods challenge conventional management ideas, and demand changes in organization structure, culture, and management practices in traditional software development organizations.

S33	Empirical Investigation on Agile Methods Usage: Issues Identified from Early Adopters in Malaysia
Author	Ani Liza Asnawi, Andrew M. Gravell, and Gary B. Wills
Year	2011, XP, Vol. 77Springer (2011) , p. 192-207
Source	Proceeding of Agile Processes in Software Engineering and Extreme Programming - 12th International Conference, Pages 192-207
Type	Proceeding

The authors conducted a qualitative study to understand the issues that are faced by early adopters in Malaysia where Agile methods are still relatively new. The initial study involves 13 participants including project managers, CEOs, founders and software developers from seven organizations.

This study has shown that social and human aspects are important when using Agile methods. While technical aspects have always been considered to exist in software development, they found these factors to be less important when using Agile methods. In this study they coded issues about education and training into knowledge. Education leans more towards understanding the concept and roles and how Agile is different from other methods. It is essential to ensure that the customers fully understand the concept or how Agile works. Agile provides opportunities for the customers, where they are also taught to relay their requirements easier. If the customers do not understand the method, it is difficult to use Agile. Apart from knowledge, mindset is equally important for the adoption factor of Agile methods. The mindset should be from the stakeholders' perspective. Agile is different in terms of its way of working and thus requires a change in mind set. In other words, those involved must be willing to change the way they work especially those who have been using different methods for a long time. All of the organizations they interviewed mentioned the importance of people's attitude. Customers, developers and people involved in Agile must understand their roles and responsibilities. From the interviews, they found that small and startup companies are more appropriate for the culture of Agile. In terms of communication, they found, agile that emphasized on communication helps to solve problem and avoid wrong assumption about the requirements.

Appendix B: Agile Expert Interview Questions

This part provides the questionnaire that was used in the interview with agile consultants.

1. What is your opinion about agile practices, what is the strongest point?
2. How is important of the involvement of business expert in agile development?
3. How does the developer team communicate with the business expert?
4. Does the developer team have direct access to the business expert?
5. Are the requirements based on business value?
6. Are the business values always clearly stated and visible to all members of team by business expert?
7. Does the developer team always understand properly the business expert needs and wants?
8. Are there metrics and project information displayed prominently via big visible chart for all stakeholders to see? So the stakeholders know what the developer team working on
9. Is there a frequent review of the development process to make it more efficient?
10. What are the factors causing the changes in a project?
11. How are the agreements between the developer team and the business expert made and accepted?
12. Are all the stake holders satisfied with the development process?
13. What are the aspects of development process that leads to stakeholder satisfaction?
14. Do the results always meet the stakeholder's expectations?
15. Are all the stakeholders satisfied with the results?
16. What is your opinion about stakeholder satisfaction degree when used Agile methods compare to Traditional methods?
17. How do you measure the stakeholder satisfaction with respect to software products that have been developed? What are the measurement instruments?

Appendix C: Web Survey

This part provides the questionnaire that was used in the web survey with its results. There were 19 persons that have filled in the web survey.

1.	Which best describes your current position?	
	a. IT manager	11 %
	b. QA/tester	11 %
	c. Developer	58 %
	d. Project manager	5 %
	e. other	16 %

2.	How many years of work experience do you have?	
	a. < 2 years	5 %
	b. 2-5 years	58 %
	c. 5-10 years	21 %
	d. 10-20 years	16 %
	e. 20+ years	0 %

3.	How many people work in the IT/systems/development department within your company?	
	a. < 11	42 %
	b. 11-50	21 %
	c. 51-100	11 %
	d. 101-500	11 %
	e. 501-1000	11 %
	f. 1000+	0 %

4.	Which sector is your organization working in?	
	a. Financial	0 %
	b. E-commerce	0 %
	c. Government	11 %
	d. IT Consultant	5 %
	e. Technology (including software)	53 %
	f. other	32 %

5.	Did you ever hear about agile software development?	
	a. Yes	79 %
	b. No	21 %

6. Did you ever work with agile software development in your projects?	
a. Yes (skip question 7, then you can continue on the next questions)	47 %
b. No (answer question 7, then you can leave this survey)	53 %

7. What is the most important reason why you don't apply agile software development?	
a. Organizational culture	33 %
b. Customer, they aren't willing to involve too much in the development process	11 %
c. Project scale, you think agile software development doesn't fit for your project	11 %
d. Staff, unavailability of staff with qualified skills	0 %
e. Never knew that about Agile software development	44 %

8. How many agile projects has your organization run?	
a. 1-5	70 %
b. 6-10	20 %
c. 11-20	10 %
d. 21+	0 %
e. We are still in the pilot phase	0 %

9. How long has your organization been using agile software development methods?	
a. < 1 year	30 %
b. 1-2 years	60 %
c. 3-4 years	10 %
d. 5-10 years	0 %
e. > 10 years	0 %

10. How much experience at agile software development do you personally have?	
a. < 1 year	30 %
b. 1-2 years	50 %
c. 3-4 years	10 %
d. 5-10 years	10 %
e. > 10 years	0 %

11. What is the agile software development method you mostly use?	
a. Scrum	90 %
b. Extreme Programming (XP)	0 %
c. Dynamic Systems Development Method (DSDM)	10 %
d. Rational Unified Process (RUP)	0 %
e. other	0 %

12. According to your experience, what is the greatest benefit obtained from agile software development?	
a. Accelerate time to market	22 %
b. Increase productivity	22 %
c. Better alignment IT-Business	33 %
d. Availability to manage changes	11 %
e. Enhance software quality	0 %
f. Decrease cost	11 %

13. What percentages of your agile projects were successful? (Successful means that the project has terminated and satisfied the stakeholders)	
a. > 91%	22 %
b. 71-90%	22 %
c. 51-70%	22 %
d. < 51%	0.00 %
e. Too early to tell	33 %

14. What is the largest team size which your organization has been successful with agile approaches?	
a. 1-5 people	33 %
b. 6-10 people	55 %
c. 11-20 people	11 %
d. 21-50 people	0 %
e. 51+ people	0 %

15. Have you ever been involved in co-located agile teams? (Co-location means that not everyone is in the same room)	
a. Yes	56 %
b. No, (skip question 16)	44 %

16. What was the success rate for co-located agile teams?	
a. > 91%	17 %
b. 71-90%	0 %
c. 51-70%	33 %
d. < 51%	0 %
e. Don't know	50 %

17. How have agile approaches affected your productivity?	
a. Much higher	11 %
b. Somewhat higher	78 %
c. No change	0 %
d. Somewhat lower	11 %
e. Much lower	0 %

18. How have agile approaches affected the quality of the systems produced?	
a. Much higher	0 %
b. Somewhat higher	78 %
c. No change	22 %
d. Somewhat lower	0 %
e. Much lower	0 %

19. How have agile approaches affected the cost of development?	
a. Much higher	0 %
b. Somewhat higher	33 %
c. No change	44 %
d. Somewhat lower	22 %
e. Much lower	0 %

20. How was the stakeholder involved in your agile projects?	
a. Very much involved	0 %
b. Involved	67 %
c. A little bit involved	33 %
d. Not involved	0 %

21. How have agile approaches affected stakeholder satisfaction?	
a. Much higher	38 %
b. Somewhat higher	38 %
c. No change	25 %
d. Somewhat lower	0 %
e. Much lower	0 %

22. (Open question) In your opinion, what are the three most important things that make agile software development successful?

23. (Open question) In your opinion, what are the three most important problems you encountered with agile software development methods?

Appendix D: Project Plan

PROJECT STATEMENT

Formal Client

The Professorship of Software Quality and Testing of Fontys ICT act as the formal client in the success of Agile Software Development – EQuA Project. The Professorship is the participant of the EQuA (Early Quality Assurance in software production) research project.

Project Leader

As the project leader is Mr. Jacob Brunekreef. He is a Software Quality and Testing researcher and teacher in Fontys University of Applied Sciences.

Current Situation

Agile software-development methods were created in response to the business community asking for lighter weight, faster, and nimbler software development processes. This is especially the case with the rapidly growing Internet software industry and mobile application environment.

However, like other methods, Agile software development methods have its own advantages and unsuitable for every situation, projects, products, and people. Agile software-development methods enable tolerance to the changes of requirements so it can be quickly addressed, but on the other hand, agile software development methods lead to decreased productivity.

Project Justification

The Professorships of Software Quality and Testing of Fontys ICT is the participant in the EQuA research project. EQuA (an acronym of Early Quality Assurance in software production) is a 4-year research project focusing on quality issues related to the first phases of software development.

Agile software development is gaining interest from both academia and industry. Although many articles and books have discussed about agile software-development methods, few of them discussed the agile methods impact on software quality and customer satisfaction. Agile supporters claim that the use of agile methods leads to higher software quality and customer satisfaction rather than the use of traditional (waterfall-like) methods. Importantly, evidence for this claim was needed. This was the starting point of this research, and resulted in the following research questions:

1. What theoretical and practical evidence can be found in literature for the claims mentioned above?
2. What practical evidence can be found in IT companies applying agile software-development methods?
3. What selection factors of agile methods instead of traditional methods are influenced by cultural aspect?

Project Product

The main product of this project is a document (Graduation thesis) with the answers on the four research question that have to be described with conclusions and recommendations.

Project Deliverables

Deliverables:

- Project plan
- Questionnaire
- Interview minutes
- Literature study
- Graduation thesis
- Graduation presentation

Project Constraints

During the project, the researcher will communicate with both Business and IT Manager of some Companies (as participants) in the Netherlands and Indonesia with the aim to obtain information regarding the implementation and results of Agile methods in the field of software development. The project is only analyzed information about Agile software development methods and do not develop software.

Project Risks

The Success of agile software development – EQuA Project will be mainly human participation associated, Business Manager and IT Manager of Company (participants). The greatest risks in this project are time and communication. Those risks could arise from participants (Business and IT manager) and/or the researcher himself.

Description	Probability*	Severity of Impact	Prevention	Response Action
Poor capture of data	3	Failure to get important information	Focus on making a good questionnaire	Re-interview
Participant refuse invitation	3	Failure to make interview appointment	Compile a list of participants as much as possible	Contact other participants
Participant is a busy person	3	Failure to make interview appointment	Make flexible appointment date and time	Give other time option to participant
Lack of language	2	Failure to get important information	Use voice recorder during interview	Use translator
Lack of time	2	Failure to make online	Use online	Send

		interview appointment	questionnaire or email to communicate	questionnaire via email
Difficulty finding participants	1	Failure to get important information from participants	Find contact person from Internet, lecturer, colleague as much as possible	Contact colleague

**On scale 1 (lowest) to 5 (highest)*

PROJECT PHASING

There are a total of 9 phases in this project:

Phase	Activities	Deliverable	Deliverable ready
Project Definition	- Drafting project plan - Meeting with project leader to discuss drafted project plan - Initiating project plan	- Project plan	- Thu 3/1/12
Research Method	- Finding scientific literature - Drafting methodology design - Meeting with project leader to discuss the best methodology that will be used - Making methodology design consider cost and time constraints	- Literature study - Methodology design	- Fri 3/2/12 - Thu 3/22/12
Questionnaire Design	- Drafting questionnaire - Meeting with project leader to discuss the drafted questionnaire - Making questionnaire	- Questionnaire	- Thu 3/29/12
Sample Selection	- Meeting with project leader to discuss participants - Contacting participants	- Potential participants list	- Thu 3/29/12
Interview	- Interviewing Netherlands participants (direct interview) - Interviewing Indonesia participants (online interview, online questionnaire, email)	- Collected information	- Tue 5/8/12
Data Entry	- Entering collected information to analyzed	- Grouped information	- Wed 5/16/12
Analysis	- Analyzing information	- Analyzed information	- Thu 5/24/12
Final Report	- Drafting final report - Meeting with project leader to discuss drafted final report - Making final report	- Final report/ Graduation thesis	- Tue 6/26/12
Presentation	- Presenting final report	- Presentation	- Thu 6/28/12

Planning of Activities

Activity	Description	Predecessor	Expected duration (day)	Finish date
1	Drafting project plan	-	5	Thu 2/9/12
2	Meeting with project leader to discuss drafted project plan	1	1	Fri 2/10/12
3	Making final version of project plan	2	5	Fri 2/17/12
4	Finding scientific literature	-	13	Fri 3/2/12
5	Drafting methodology design	4	6	Mon 3/12/12
6	Meeting with project leader to discuss methodology design	5	1	Tue 3/13/12
7	Making methodology design	6	6	Wed 3/21/12
8	Drafting questionnaire	4	6	Mon 3/12/12
9	Meeting with project leader to discuss drafted questionnaire	8	1	Tue 3/13/12
10	Making questionnaire	9	11	Wed 3/28/12
11	Meeting with project leader to discuss potential participants	-	1	Fri 4/6/12
12	Contacting participants	11	11	Mon 4/23/12
13	Interviewing participants (gathering information)	-	16	Fri 5/11/12
14	Entering information	13	6	Mon 5/21/12
15	Analyzing information	14	6	Tue 5/29/12
16	Drafting final report	-	11	Fri 6/8/12
17	Meeting with project leader to discuss drafted final report	16	1	Mon 6/11/12
18	Making final report	17	1	Tue 6/26/12
19	Presenting final report	18	1	Wed 6/27/12

MANAGEMENT PLAN

Money

The success of Agile Software Development – EQuA Project's cost throughout the duration of the project:

- Travel expenses to the interview place in Netherlands (train, bus, tram)

Skills

Skills needed for this project are:

- Good writing and communication skill
- Good analytical skill
- Basic knowledge in Agile Software Development methods

Quality

The quality of this project depends on the answer of the research question with the conclusion and recommendations, also the clarity and completeness of the project documentation.

Information

	Project plan	Methodology Design	Questionnaire	Contact Person Information	Data	Final Report
Formal Client	R	-	-	-	-	R, Gf, A
Project Leader	R, Di, Gf, A	R, Di, Gf, A	R, Di, Gf, A	R, Di, Gf, A	R, Di, Gf, A	R, Di, Gf, A
Researcher	Wo, S, Di	Wo, S, Di	Wo, S, Di	Wo, S, Di	Wo, S, Di	Wo, S, Di

Legenda:

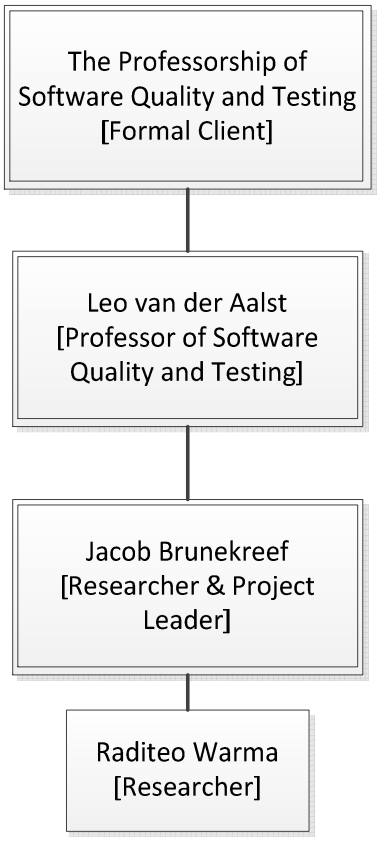
R	Receive	Gf	Give feedback	Wo	Work on
Di	Discuss	S	Send	A	Approve

Time

The success of Agile Software Development – EQuA Project starts at February 3rd, 2012 and will last for 20 weeks long until June.

Organization

Below is a chart that represents the organization of the success of Agile Software Development – EQuA Project


The Professorship of Software Quality and Testing [Formal Client]
Leo van der Aalst [Professor of Software Quality and Testing]
Jacob Brunekreef [Researcher & Project Leader]
Raditeo Warma [Researcher]