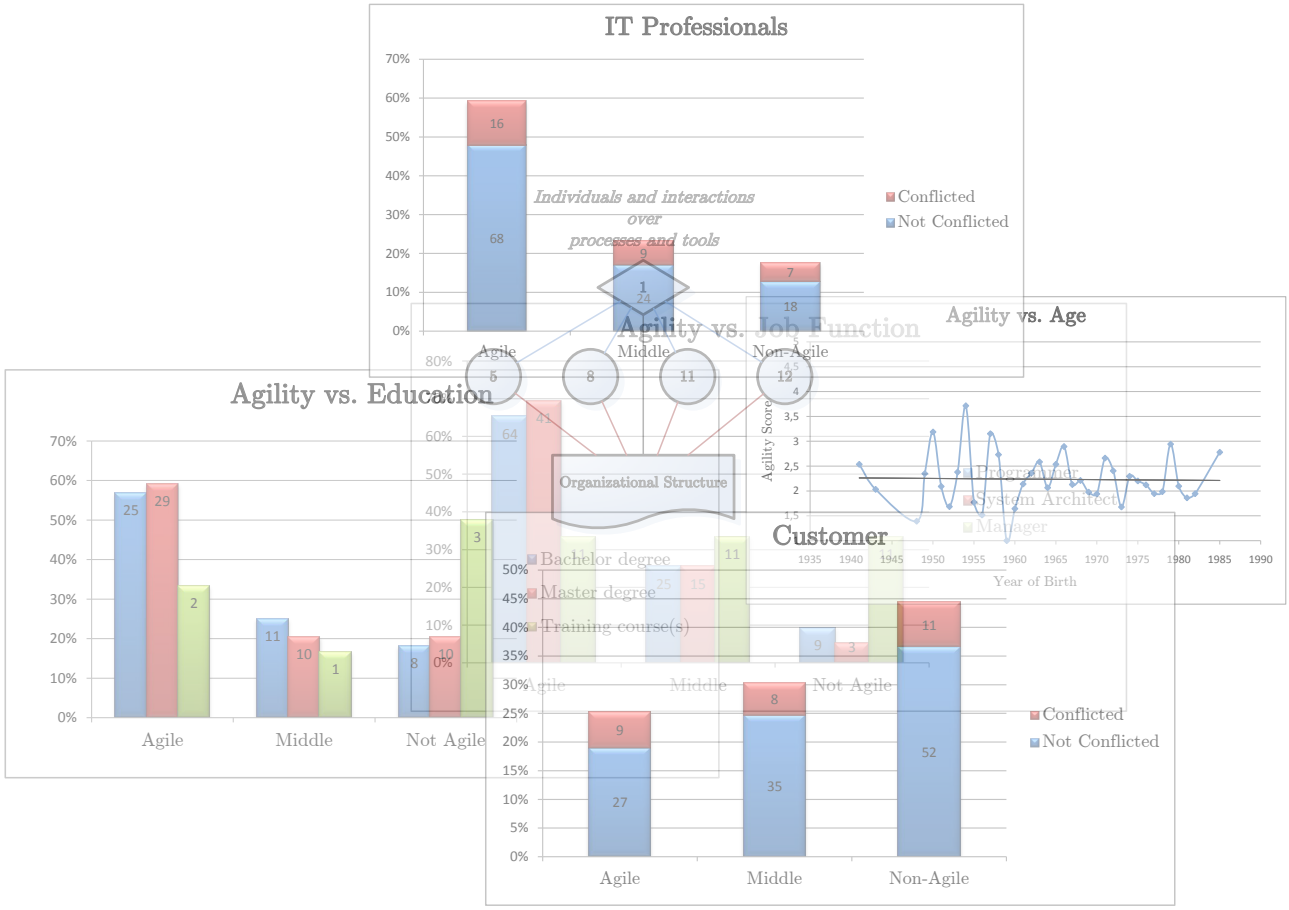
Measuring Agility

A Quantitative Survey Among IT Professionals



Group: d614a
Rasmus Mortensen
Taregh Jasemian
Thomas Boje Nielsen

Aalborg University
Department of Computer Science, Fredrik Bajers Vej 7E, DK 9220 Aalborg Øst

Title:
Measuring Agility: A quantitative survey among IT professionals

Research Field:
Information Systems - Systems Development

Project period:
Software Engineering 10th semester, Feb. 1^{st} - June 8^{th}, 2007

Project group:
d614a, E3-113

Group members:
Taregh Jasemian
Rasmus Søren K. Mortensen
Thomas Boje-Nielsen

Supervisor:
Andreas Munk - Madsen

Circulation: 6

Page count:

Report: 118

Appendices: xiii

Total: 147

Abstract:
Several studies indicate an increase in the use of agile methods and practices in the software industry and that these methods and practices bring value to development projects. However, the few quantitative studies which examine the extent to which Agile Software Development is actually used, often rely on practitioners use and knowledge of specific agile practices and methods. Without distinguishing true use from hype-related use, such studies can be questioned. This report describes how the Agile Manifesto and its principles are operationalized into an instrument for measuring agility by assessing attitude and how the instrument is used in a quantitative survey among IT professionals. The project is based on an iterative survey research model and embody the design, execution and analysis of a questionnaire in order to answer four research questions. These concern the agility of IT professionals, teams, organizations and customers, and a software projects suitability for Agile Software Development.
The project concludes that a large part of Information Technology (IT) professionals and their teams are agile-minded and that most software projects are suited for Agile Software Development. Organisations and customers on the other hand are not perceived as equally agile.

Taregh Jasemian

Rasmus S. K. Mortensen

Thomas Boje-Nielsen

Aalborg University, June 8^{th}, 2007.

Preface

This report is completed on the 10^{th} semester in Software Engineering in the period between Feb. 1^{st} and June 8^{th} 2007 by group d614a at Aalborg University, Department of Computer Science.

Report Outline

The report documents the master's thesis *Measuring Agility: A quantitative survey among IT professionals*. The report primarily centers around a survey conducted among IT professionals in Denmark. It also explores Agile Software Development and operationalizes it in order to construct an instrument for measuring agility by assessing attitude.

Chapter 1 - Introduction: Introduces the research questions and Agile Software Development and presents the research model.

Chapter 2 - Related Research: Describes and presents the findings of a literature review of existing surveys and articles which deal with Agile Software Development and solutions for measuring agility is examined.

Chapter 3 - Operationalization of Agility: Takes the information drawn from the review and unravels the principles and values behind Agile Software Development so they can be used to measure agility.

Chapter 4 - Constructing the Instrument In this chapter the results from the operationalization is used to constructs the agility measurement instrument and questionnaire. The chapter also elaborate on the theory behind reliability and validity and discusses questionnaire administration.

Chapter 5 - Analysis Concerns the reliability and validity of the survey, presents the analysis and answers the research questions.

Chapter 6 - Discussion This chapter discusses the answers to the research questions and reflects on the lessons learned.

Chapter 7 - Conclusion Concludes on the survey, research method and process.

Appendices Presents the resume in Danish and the questionnaire in its entirety and finally an article published in the June edition of the magazine ProsIT.

Acknowledgments

In relation to this project we would like to give thanks to our project supervisor, Andreas Munk-Madsen, who guided us during the project and Ivan Aaen and Clive Sanford for their assistance in the questionnaire design process. Also special thanks to PROSA, IDA, IKT-Forum, Teknologisk Institut and Rambol for their help with distributing the questionnaires. Finally we would like to give thanks to the persons who helped test the questionnaire and all the respondents who participated in the survey.

Contents

1 Introduction

This chapter concerns the choice of subject and motivation behind the research. Next, the research questions are presented, basic concepts are clarified and the applied research method is introduced.

1.1 Choice of Subject

Agile Software Development is one of today's hot topics within the software community. As a software development methodology its ideas and elements are debated heavily among practitioners, methodologists and researchers, with advocates on each side. The reason for its (un)popularity is partly because it breaks with the tradition of controlling and planning everything through process and instead centers on people.

1.1.1 Motivation

Since the term "Agile Software Development" was established in 2001, the interest in the methodology has been increasing as Figure 1.1 illustrates.

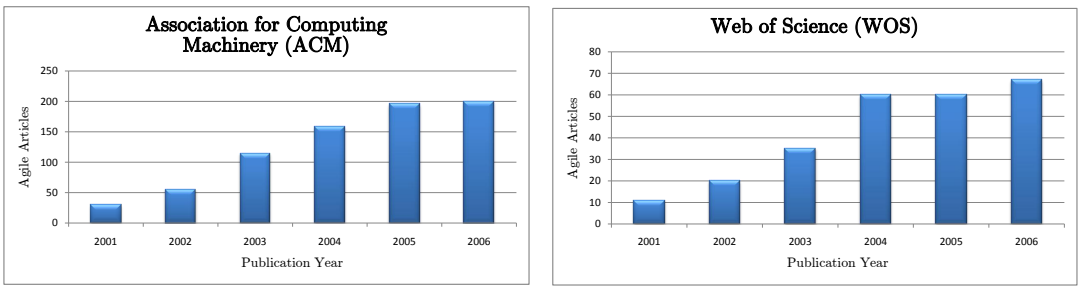


Figure 1.1: The increase of agile articles published in the years between 2001 and 2006, found on two online journal databases.

The figure shows the increasing number of articles published each year since 2001 on two journal databases and containing the keywords *"Agile"*, *"Software"* and *"Development"* in either the title, abstract or keyword list of the article.

A study of over 23000 software projects, done by the Standish Group in 1998, listed the most important factors leading to software project success [37]. The top of the list includes several factors which focus on people aspects in software development such as *Customer involvement* and *Executive support*. Agile Software Development is considered a very people-centric methodology as it emphasizes the need for collaboration and communication. It also specifically addresses a number of the factors on the Standish success-criteria list: *Customer involvement*, *Small milestones*, etc. In addition, many research studies across the world have shown that the use of Agile Software Development has indeed had a positive impact on software projects. This is for instance seen in the form of added business value, lower defect rates and decreased time-to-market [26] [14] [36]. Given that value in the agile practices and ideas exists, the subject is an important and interesting research field within today's software community.

Despite the interest and strong indication of value in the agile approach, there has not been any research on the current state of Agile Software Development in Denmark. A few studies have been conducted in other parts of Europe and the United States, all of which show an increase in the knowledge and use of agile practices and ideas [31] [36] [14].

1.1.2 Focus

This research project focuses on what Agile Software Development is and what it means to be agile. Highsmith [4] states that it is not merely a methodology but uses the word "Ecosystem" to communicate how Agile Software Development involves the entire organization and its surroundings. As the agile approach also brush off many of the traditional ideas of software development, it is clearly a subject where opposing values and interests of organizations, teams, developers and customers can appear. Williams and Cockburn [28] note that the power structure differs from organization to organization. In some organizations, developers might take many key decisions, including business related, while the customer is more reluctant to participate. In others, control might be valued and managers would be expected to exercise their control. As Agile Software Development among other things involves a spread of decision making authority, its ideas would clearly relate differently to different types of organizations, teams, developers and customers.

Boehm and Turner [15] explain that project characteristics matter when discussing the applicability of agile and plan-driven methods. While some projects may receive benefits by using agile ideas and practices, other projects would experience the opposite and hence require a more planned approach. They state that an agile home-ground exists for Agile Software Development and a plan-driven home-ground exists for plan-driven methodologies.

1.2 Research Questions

Based on our motivation for and focus in the subject, four research questions are formulated:

- *How agile are IT professionals?*
- *Is there a difference in the degree of agility of IT professionals and the agility of their team, organization and customer?*
- *Is there a correlation between IT professionals' agility and their characteristics, i.e. their age, job function, experience and education?*
- *Is software projects' context suited for Agile Software Development?*

The above-mentioned questions are answered in this report through a quantitative research survey among IT professionals in Denmark.

1.3 Agile Software Development

The term "Agile Software Development" was coined in 2001 when 17 people, now known as the authors of the Agile Manifesto, met at a ski-resort in Snowbird, Utah.

A predecessor to Agile Software Development and a central part of agile methods is the Iterative and Incremental Development (IID) approach. The following section looks at some examples on IID projects through the years as described by Larman [8].

1.3.1 Iterative and Incremental Development

In the early days of software development no real development model existed, except a code and fix approach. Later on the plan-driven or waterfall model was created, which would dominate the approach to software development from the 1970's and onward. The waterfall model in essence builds on a sequential series of phases, such as requirements analysis, design, implementation, and testing. It relies on documentation of the entire system before any code is written and does not allow for iterations. As a contrast to this IID relies on incremental builds along the lifespan of a project. In many ways IID is a precursor to what we today call agile methods and consists of e.g. the evolutionary and spiral models. [8]

In his book, *Agile and Iterative Development* [8], Larman unravels the history of IID by examining a number of examples on IID projects. It was first used as a method for developing software on NASA's Project Mercury from 1961-63, where some of the personnel later seeded the Federal Software Devision (FSD) at IBM. One of the first major projects for FSD was the development of a life critical system for the USA Trident submarine in 1972, where IID was noted as one of the main success factors. Though examples of such projects continue well into the 1980's, the waterfall model was still the model of choice in the software industry. [8]

In the early 1980's an official Department of Defence (DoD) standard for development was released because of a disturbing amount of software project failures. The standard named DOD-STD-2167 build upon the waterfall model. The shortcomings of this was soon clear as the standard was updated in the late 1980's to include a more IID friendly approach. However, the new standard 2167*A* still had the single step diagrams of the waterfall model and still relied on a document-driven approach. This led to a further update of the official DoD development standard called MIL-STD-498 in 1994 and DoD 5000.1 in 2000, which specifically describes the evolutionary and spiral models. However the 2167 and 2167*A* standards have had a great deal of influence over other standards developed outside the US, which still to this day rely on the waterfall model and a document-centric approach. [8]

1.3.2 Lightweight Methods

Some software development practitioners were dissatisfied with the burden of documenting everything in the smallest detail and the inflexibility of the development process. They felt that the software development methods available put too much emphasis on process as opposed to people and product [8]. This lead to the creation of a new breed of software development methods, dubbed "lightweight methods" because of their simple approach to development. These methods included eXtreme Programming (XP) and Scrum, whose straightforward concepts of iterative planning and development, coupled with only a few simple guidelines or practices, signaled a shift away from the heavyweight methodologies. XP for example consists of only 12 practices to follow. The lightweight methods sought to put people in the center of development, by strengthening the contact between developers and customers, and by acknowledging that developers were the most important asset to any software project. However, the methods and practices was often criticized for encouraging cowboy coding, and just for people who wanted an excuse to start coding and not bother with design and documentation. Nevertheless, the lightweight methods provided the software community with a formalized alternative to the heavyweight methods.

1.3.3 The Agile Manifesto

To create a common set of values and principles that span over the different lightweight development methods and thus bringing these together, a number of lightweight methodology practitioners met at Snowbird, Utah. The view shared among the participants, was that important ideas and approaches to software development *"...had been stifled and not been treated seriously enough, particularly by people interested in software processes."* [34].

The meeting gave rise to a number of important items concerning Agile Software Development:

- A Manifesto for Agile Software Development.
- The creation of 12 agile principles.
- The founding of The Agile Alliance.

- The umbrella name *"Agile"* for all the lightweight methods, following the Agile Manifesto values and the 12 agile principles.

Contrary to the many methods and practices that focus on different aspects of Agile Software Development, the Agile Manifesto and the 12 principles tries to define the entire field of Agile Software Development. Together they describe the values and principles behind its philosophy and what methods and practices must achieve to be considered agile.

The Manifesto states:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> ***Individuals and interactions*** *over processes and tools*
> ***Working software*** *over comprehensive documentation*
> ***Customer collaboration*** *over contract negotiation*
> ***Responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*[33]

Besides the four valued statements, it is important to notice the use of the word *"uncovering"* in the first sentence. As explained by the authors, the word *"uncovering"* illustrates that the ideas that Agile Software Development builds on, are neither new nor should they be seen as the silver bullet to software development [29]. Furthermore, the last sentence is important as it clearly states that Agile Software Development does not abandon the items on the right, but simply values the items on the left more.

The 12 agile principles that initially was developed at the meeting and have evolved since, are examined and discussed in more detail in Chapter 3.

1.4 Research Method

The applied research method in this project is *survey research*, where a questionnaire is used as a data collection tool. *Survey research* is one of the most important quantitative research methods used within in social science [41]. Therefore disciplines and techniques within the social science field are applied to design our research method. This involves research disciplines and techniques explained by Kumar in *Research Methodology* [6] and Trochim on *Web Center for Social Science Research Methods* [41].

This section explains the overall research method used to answer the research questions and describes the distribution of the 9^{th} and 10^{th} semester activities, which together have resulted in this report.

1.4.1 Research Process Model

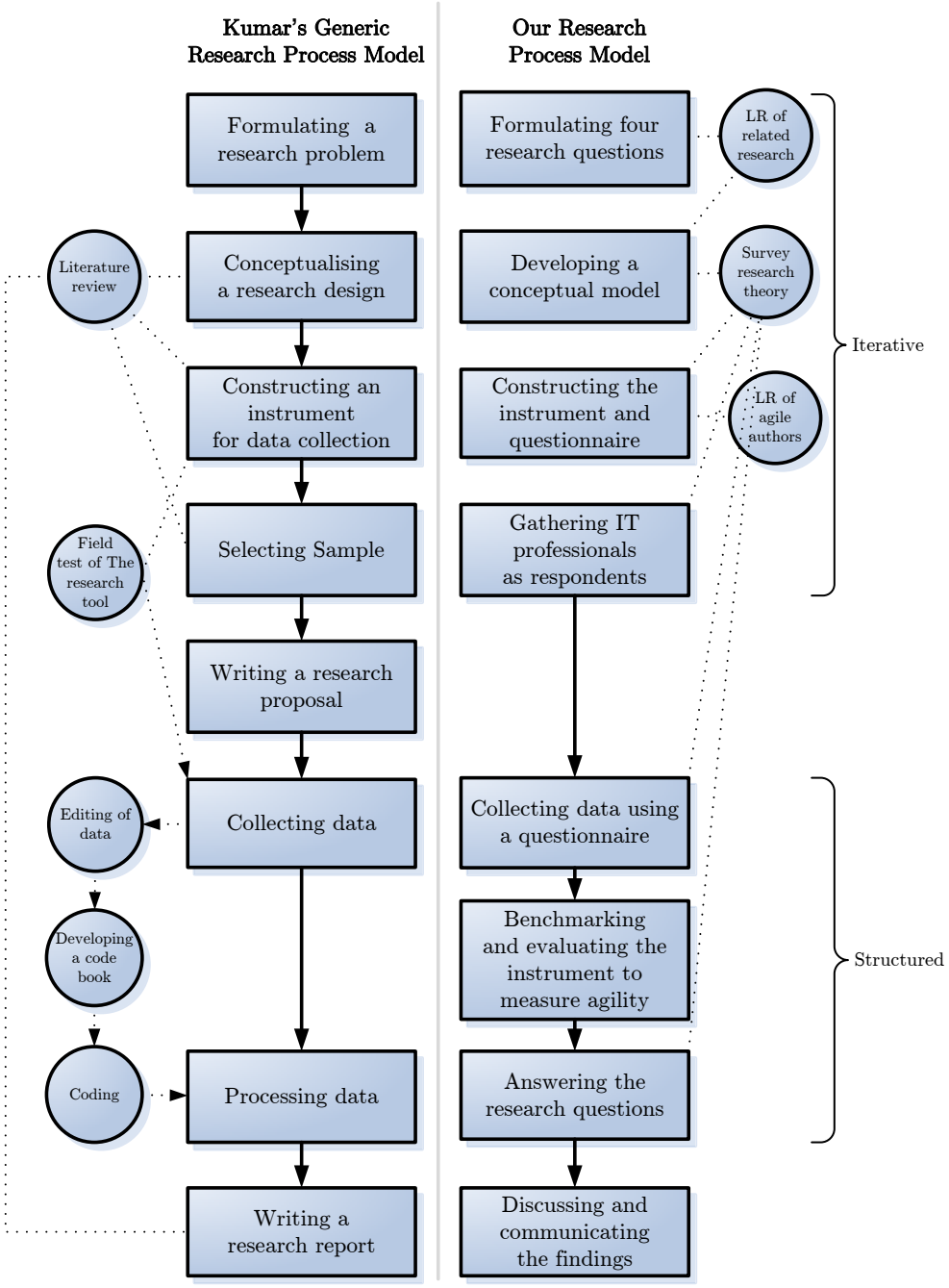


Figure 1.2: Kumar's generic process model compared with our research process model.

Kumar notes that research methodologies may vary in their substance and content, but that their broad approach to inquiry is similar [6]. He proposes an eight staged generic research process model, which can be applied to a number of disciplines within social science. Kumar emphasizes that it is not strictly necessary to follow the model in the proposed sequence. Thus it is used as a guideline and as inspiration for the development of our own research

model. In Figure 1.2 both models are paired with each other for comparison.

Kumar's generic model is very sequential as each step is an operational task that needs to be followed in order to conduct a research. However, some iterative measures are introduced into the model by the intermediary steps represented by the circles. E.g. the intermediary step *"Field test of research tool"* indicates that first a prototype of the instrument is constructed, then a sample is selected and data is collected after which step 3 is visited again in order to tune the instrument and so forth.

Our research process model consists of eight steps where most reflect the steps proposed by Kumar. Though our process is much more iterative than Kumar's generic model, the term "step" is still used to describe the activities. The first five steps are not carried out in a sequential order but done in parallel. The point of no return from which the steps are carried out sequentially is when the data from respondents are collected and processed. In addition, step five in Kumar's model, *Writing a research proposal*, concerns writing a proposal for the study in order to have it approved by committees, which is not necessary in this project. Each of the steps in our model will now be explained

Step One – Formulating Four Research Questions

The first step in the research process model involves acquiring knowledge of the research problem domain and formulation of the four research questions. In order to establish reasonable and valid research questions, existing research on Agile Software Development need to be examined. This is indicated by the intermediary circle activity of literature review(LR) of related research. This activity also takes place in step two and illustrates how the different activities involved in step one and two progress iteratively. The literature found and reviewed helps clarify the problem domain, which again assists in finding more literature of increasing relevance.

Step Two – Developing a Conceptual Model

The second step involves a review of existing literature and the development of a conceptual model of the 9th and 10th semester activities. In this step, existing research is examined in order to draw inspiration from related studies on applied methods and to establish knowledge on research designs. This is illustrated by the intermediary circle activity of literature review of related research. To develop a conceptual model we have to look at theory behind survey research. The survey theory is also used in other steps as illustrated in the model. The output of this step is the conceptual model seen in Figure 1.3.

Step three – Constructing the Instrument and Questionnaire

The third step involves the construction of an instrument to measure agility and the design of the questionnaire which uses the instrument. The instrument should measure the agility of four different perspectives; IT professionals, teams, organizations and customers. The first perspective, i.e the agility of IT professionals, is the primary objective of the instrument and

therefore more attention should be paid to this than the other three perspectives. The construction of the instrument requires a thorough examination of Agile Software Development in order to establish ample knowledge for operationalization of agility. The questionnaire implements the measurement instrument and must be designed so that additional and sufficient data for answering the research questions can be collected and the results can be validated. Inspiration from related survey reports and articles assists in the design of the questionnaire questions. Boehm and Turners five critical agility and plan-driven factors [15] are transformed into questionnaire questions in order to answer the fourth research question. The design of the questionnaire also involves the application of known questionnaire design techniques described by Trochim [41] and Kumar [6]. The needed information and knowledge to construct an instrument, a literature review of the Agile Manifesto authors are conducted. Besides that information from the survey research theory is used to construct and design the questionnaire. This is illustrated by the two intermediary circle activities connected to the this step.

Step Four – Gathering IT Professionals as Respondents

Step four is crucial in order to answer the research questions. Lack of respondents will result in an insufficient amount of data to draw valid conclusions. The different abilities of questionnaire administration approaches as described by Kumar [6] are examined and the most appropriate is selected. A satisfying sample of respondents must be established, either through assistance from external organizations and companies or by contacting respondents directly on e.g. conferences or networks. Step six in our model requires that two different samples are set up. The first sample can be described as the main sample and should represent the field of IT professionals as closely as possible. The second sample should consist of respondents who can be considered to be agile, as this sample is used to benchmark and evaluate the instrument. To address the issue of sample and gathering respondents the information from survey research theory is used, hence the intermediary circle that is connected to this step.

Step Five – Collecting Data Using a Questionnaire

Step five constitutes the point of no return. When the questionnaire, containing the finished instrument for measuring agility, is constructed and the two samples of respondents are established, the questionnaire is distributed and data is collected. Again information from the survey research is used in this step.

Step Six – Benchmarking and Evaluating The Instrument to Measure Agility

In the sixth step, data collected from the sample of agile respondents is used for benchmarking. The benchmark identifies a cutoff point for agility and is also used to evaluate the reliability of the instrument. When the the cutoff point for agility is identified it is used when processing the data for the main sample of respondents and answering the research questions.

Step Seven – Answering the research questions

Step seven in the model involves processing and analysis of the collected data in order to answer the research questions. As electronic tools are used for data collection, the intermediary steps *Developing a code book* and *Coding* presented in Kumar model are not followed. However, the data set is edited according to the procedures described by Landau et. al. [7]. The edited data set is then analyzed and its validity is examined. After these preliminary tasks, the data is subjected to a principle components extraction and analyzed in order to answer the research questions. In this step information from the survey research method is also used, as indicated by the intermediary circle activity connected to this step.

Step Eight – Discussing and Communicating The Findings

The last step of Kumar's process model is writing the research report. It is our intention to write a large part of the research report in parallel with other the other activities. Therefore the last step in our model is the discussion of the findings, lessons learned and communication of results to external partners and stakeholders. This involves an article for PROSIT describing the relationship between the agile methods, practices, values and principles. This article is seen in Appendix C.

1.4.2 Conceptual Project Design Model

Figure 1.3 depicts our conceptual model which is based on Peter Checkland's conceptual model building as it is described by Rose [27]. Our conceptual model depicts the activities for both the 9^{th} and 10^{th} semester and the logical dependencies between the activities. An activity is depicted as a cloud and logical dependencies as arrows between the activities. The big arrow depicts transformation processes and the big arrow with text inside depicts an artifact.

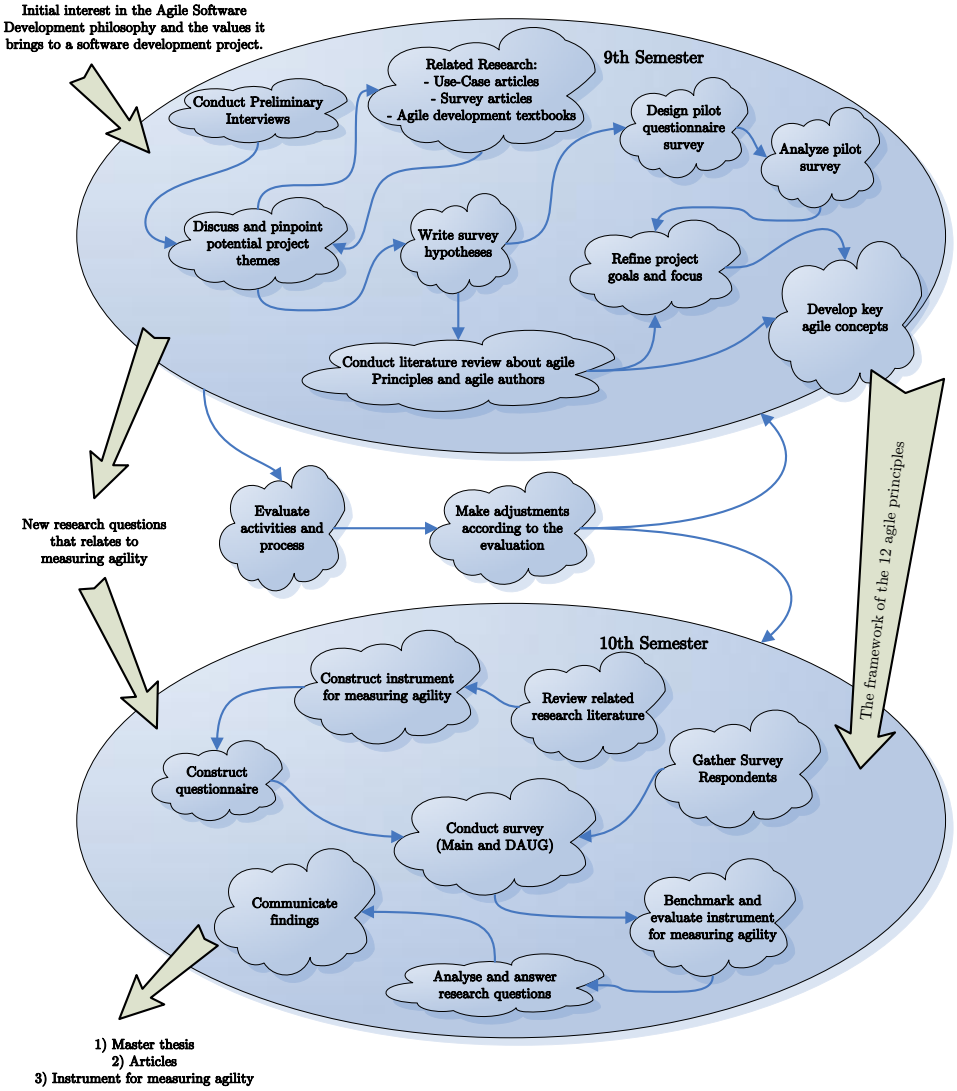


Figure 1.3: Conceptual model for our research project.

The 9^{th} semester was used to obtain preliminary theoretical knowledge on survey research and on Agile Software Development. A pilot survey among computer science students at Aalborg University was conducted in order to achieve skills and experience with survey research. Through the examination of Agile Software Development we developed a framework for reviewing the 12 agile principles. This framework was used and developed further in the 10^{th} semester.

The 10^{th} semester start with the gathering of respondents, which progresses concurrently with the review of related literature. Gathering respondents is a slow process where much of the time is spent waiting for reply from different organizations. Therefore the review of related literature fits well with this activity. The instrument for measuring agility is embedded within the questionnaire and hence the construction of the questionnaire is depended on

the instrument. When both are developed and ready and the respondents are gathered the survey is conducted. The survey is conducted on two different samples. One is the "main" sample and the second is the Danish Agile User Group (DAUG) sample of agile respondents. The DAUG sample is used to benchmark agility and evaluate the measurement instrument and the benchmark (cutoff point) is used to perform the analysis of the collected "main data" and answer the research questions. The results are then communicated in the form of this report, an instrument for measuring agility and an article.

Dependencies in the model does not universally prohibit iterative development. E.g. parts of the questionnaire does not depend on the instrument for measuring agility. The dependency merely illustrates that the questionnaire can not be completed before the instrument is completed.

1.5 Clarification of Concepts

A number of concepts used in this report can also be found in general software development literature, but with various meaning. To avoid confusion the concepts are defined here.

1.5.1 Agility

Merriam-Webster [35] defines *"agility"* as *"the quality or state of being agile"*. Continuing, Merriam-Webster defines *"agile"* as:

1. *"Marked by ready ability to move with quick easy grace <an agile dancer>"*
2. *"Having a quick resourceful and adaptable character <an agile mind>"*

The definition fits very well with the description of agile methods and the principles of Agile Software Development. Indeed agile methods should uphold the ability to provide quick progress and be adaptable. Agility, however, is not only used within software development. It is also used as a term in manufacturing and management [20]. Here it more or less is used to refer to the same attributes as defined by Merriam-Webster and have many parallels to Agile Software Development ideas and thinking. However, to avoid confusion, in this report the term is strictly used in relation to Agile Software Development.

1.5.2 Methodology

Methodology is build from the Greek word *"methodos"*, meaning *"following after"* or *"pursuit"*. The word *"ology"* means *"the study of"*. In Merriam-Webster [35] *"methodology"* is defined as *"a body of methods, rules, and postulates employed by a discipline."*.

Method and *methodology* are widely used as synonyms but in this report a distinction is made. A *method* is seen as a specific set of actions or steps on how to build software. *Methodology* is the study of one or more methods. Thus methodology is seen as a set

of recommended practices that can encompass several different methods within a specific paradigm, e.g. agile. This view on method versus methodology entails that the field of software engineering has many methods but few methodologies. [35]

1.5.3 Method

The word *"method"* stems from the Greek word *"methodos"* defined by the Merriam-Webster dictionary [35] as *"a way, technique, or process of or for doing something"* [35]. The use of *"method"* in this report correlates both to a *research method* and a *software development method*.

A *research method* is a way of conducting research and collecting empirical data.

In the field of software engineering, a *method* describes a way of designing, implementing and testing a software system. The term *"agile method"* in software engineering is used in relation to a specific development method within the Agile Software Development methodology, e.g. XP, Scrum, Crystal, etc. In this report an agile method should thus not be confused with an agile practice, such as pair-programming.

1.5.4 Practice

Merriam-Webster [35] defines the term *"practice"* as *"actual performance or application"*, e.g. *"ready to carry out in practice what they advocated in principle"*. However, this is not how we use the term practice.

In this report the term *"agile practices"* is used to describe the underlying elements that an agile method builds on. E.g. XP outlines a number of practices to follow, in order to use XP as a software development method.

1.5.5 Principle

The term *"principle"* is used often in the report and in the researched material. Most noticeably are the 12 principles of Agile Software Development. The definition found in Merriam-Webster [35] and used in this report states that a *principle* is: *"a comprehensive and fundamental law, doctrine, or assumption"*.

Therefore the relation between *agile principles* and *agile practices* are that an *agile principle* is a fundamental and underlying *law or doctrine* of Agile Software Development, that *agile practices* try to accomplish.

1.5.6 Conceptual Model

In Figure 1.4 the relationship between the different agile concepts used in this report is illustrated. The figure shows that the agile methods all have the 12 agile principles as a

common base, and each of the agile methods consist of a number of agile practices.

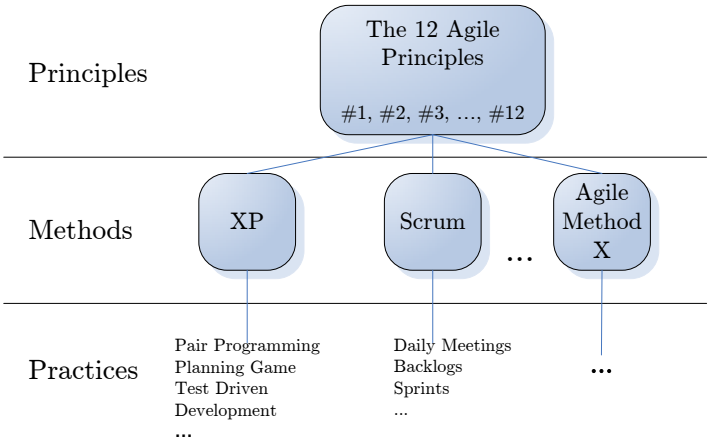


Figure 1.4: A model showing the relationship between principles, methods, and practices in the report.

2 Related Research

Kumar [6] states that a review of related research is helpful as it brings clarity and focus to the research problem, broadens ones knowledge base in the research area and improves the methodology used. In this chapter a review of research related to the subject of Agile Software Development is conducted. First the search for literature is described and then the findings are explained. The knowledge obtained through the review will guide the operationalization of Agile Software Development and assists in developing the measurement instrument.

2.1 Literature Search

Marzyck et al. [9] explains that the value and importance of a well conducted literature review cannot be underestimated. Therefore a structured search for related research articles is conducted using three online journal databases; Web of Science (WOS), Institute of Electrical and Electronics Engineers (IEEE), and Association for Computing Machinery (ACM). WOS covers approximately 850 thousand indexed journal articles and has advanced search tools that can be used to refine search results [39]. ACM has 1.4 million records in their database [30]. IEEE digital library contains more than 1.2 million documents from IEEE and IEE journals [32].

The three databases are searched for articles containing words in either the title, abstract or keyword list, that relate to Agile Software Development and surveys. The search results are limited to articles within the years spanning from 2001-2007, as the term "Agile Software Development" was created in 2001. In addition the search tool on WOS refined the search to the following computer science subject categories; *Theory & Methods*, *Software Engineering*, and *Information Systems*.

2.1.1 Results

In Table 2.1 the search criteria used and the number of hits for each search on each of the three online journal databases are shown.

Keywords	WOS	IEEE	ACM
"agil⋆"	461	323	1519
"agil⋆" AND "survey⋆"	**11**	**28**	478

Table 2.1: The number of hits for each search criteria on each journal database.

The star (⋆) is a wildcard, which indicates that the ending of the words *agil* and *survey* are arbitrary. The wildcard is used to represent that words like; *"agile"*, *"agility"*, *"survey"*, *"surveying"*, etc. all are accepted keywords.

The first search using the search criteria *agil⋆* on the journal databases gives a large number of hits. To reduce the amount of articles and to find those that related to our research area, the keyword *survey⋆* is added to the search. The keyword *survey* is selected as it narrows the hits to articles which used a research method similar to our own. This results in a manageable amount of articles, for WOS and IEEE, however, there is still too many hits on ACM (478). The list of articles found on ACM are sorted according to searched keyword relevance and the first 20 articles are chosen.

The numbers in bold in Table 2.1 and the first 20 articles from ACM are manually browsed and the most relevant are put aside for later review. Relevant refers to articles which to a more or less extent operationalize Agile Software Development for use in their study.

2.1.2 The Literature List

After manually browsing the articles that are found on WOS, IEEE, and ACM, the list is composed of seven articles. In addition to these, relevant articles and survey reports found during the course of our research and not included in the structured search are added to the literature list. These items are found through the Google Scholar search tool. The complete list of articles is seen in Table 2.2.

The list consists of nine articles and three survey reports. The specific objectives of the articles found and the specific type of research they conduct differs. Eight items on the literature list relies exclusively on questionnaires to gather data. The primary objectives in these eight include; identifying benefits and drawbacks of using Agile Software Development, identifying the largest barriers for using or adopting Agile Software Development, and describing how many and who are doing Agile Software Development. The case studies are mostly concerned with the use of a specific agile method. The objectives of the case studies are to measure the outcome and adherence to methods and in detail describes how the specific practices of the method are implemented. One article differs from the rest as it describes the findings of a workshop, where a group of experts and practitioners discussed

Title	Publisher	Source	Type	Data Collection Method	Methods & Practices	Principles & Values	Project Context Factors
Field Experiences with eXtreme Programming: Developing an Emergency Response System	Journal of ManagementInformation Systems	WOS	Article	Action Research, questionnaire	✓	✓	✓
How Agile are Industrial Software Development Practices?	The Journal of Systems and Software 79, Elsevier	WOS	Article	Interview		✓	✓
Investigating the extreme programming system - An empirical study	Springer Science + Business Media	WOS	Article	Questionnaire	✓	✓	✓
Survey Says: Agile Works in Practice	Dr. Dobb's Journal	WOS	Article	Questionnaire	✓		
How Good are Agile Methods?	IEEE Software	IEEE	Article	Questionnaire	✓		✓
Project Management in Agile and Plan-driven Companies	IEEE Software	IEEE	Article	Questionnaire	✓		
Motivations and Measurements in an Agile Case Study	Journal of Systems Architecture 52, Elsevier	ACM	Article	Observation, Interview, Questionnaire	✓		✓
An Empirical Study of Selecting Software Development Life Cycle Models	IOS Press	Google Scholar	Article	Questionnaire	✓		
Empirical Findings in Agile Methods	Springer-Verlag Berlin Heidelberg	Google Scholar	Article	Expert Discussion Group	✓	✓	✓
Digital Focus - Agile 2006 Survey Results	Digital Focus	Google Scholar	Survey Report	Questionnaire	✓		✓
Shine Technologies Agile Methodologies Survey Results	Shine Technologies	Google Scholar	Survey Report	Questionnaire	✓	✓	
Version One - The State of Agile Development	Version One	Google Scholar	Survey Report	Questionnaire	✓		✓

Table 2.2: The finale list of reviewed articles and surveys.

Agile Software Development.

The search and the review is done with respect to the needs of the project, where one objective is to collect inspiration on how to operationalize Agile Software Development, so that the agility of IT professionals can be measured. Another objective is to identify important context factors which influence the applicability of agile methods. With this in mind, the review of the nine articles and three survey reports are done and grouped into three areas; *Methods and Practices*, *Values and Principles* and *Project Context Factors*.

These three areas are used as a framework to categorize each article's use and handling of Agile Software Development. This is shown Table 2.2 and the checkmark indicates what the different articles and survey reports use and handle Agile Software Development.

2.2 Methods and Practices

The review of the related research indicats that especially survey reports had a tendency towards defining Agile Software Development on the basis of agile methods and practices. This section will examine how agile methods and practices are used in related research from the viewpoint that adherence to agile methods and practices can be used to measure the agility of IT professionals.

In a survey done by Digital Focus [31], one of the objectives is to find out *"Where agile development has taken root?"*. The question can be seen as two-dimensional. It involves finding out if Agile Software Development is being used (first dimension) and finding out what context it is being used in (second dimension). When addressing the first dimension of the question, i.e. finding out if Agile Software Development is used, respondents are asked *"Why are you interested in agile software development?"*. The question is closed-ended, meaning that a set of answers are provided for the respondent. Two of these answers read: *"Adopting/using agile practices for a project"* and *"Adopting/using agile practices companywide"*. Positive answers to either of these two are interpreted as confirmed adoption of Agile Software Development.

The main problem with this assessment is that the use or adoption of a single agile practice is treated similar to using a full-blown agile method. In Digital Foucs' survey, respondents who are trying out (or even thinking about trying out) the *Test-first* practice of XP, are put side by side with teams following XP to the letter. There is no distinction between the two. If the use of practices and methods are considered as indicators for the use of Agile Software Development, using a few practices is not enough to be considered agile. Also it is good software development practice to try out different techniques and practice occasionally to learn about them and to decide weather they work or not. Such behavior have nothing to do with *agile taking root*, but a sound software development practice.

2.2.1 Agility and Adherence Metrics

Layman et al. [23] conducts a case study on a project, that goes into a bit more detail than Digital Focus and similar survey reports, as it attempts to measure adherence to the agile

method, XP. However, it still uses the agile practices as described in XP as a basis for, what it means to do XP and thereby being agile.

For this kind of case study, the researchers have developed a certain framework; The *XP-evaluation framework*. The framework contains three groups of metrics for retrieving data; *Context Factors*, *Adherence Metrics* and *Outcome Measures*. The *Context Factors* and *Outcome Measures* contains metrics for gathering the project context data (type, size, people, etc.) and project outcome data (defect rate, Lines Of Code (LOC) / developer, etc.) respectively. However, for assessing peoples agility the *Adherence Metrics* are interesting to examine. The purpose of these metrics are to *"... to express concretely and comparatively the extent to which a team follows XP"* [24].

The idea of measuring conformance to agile practices or methods in order to say something about peoples agility is illustrated in Figure 2.1.

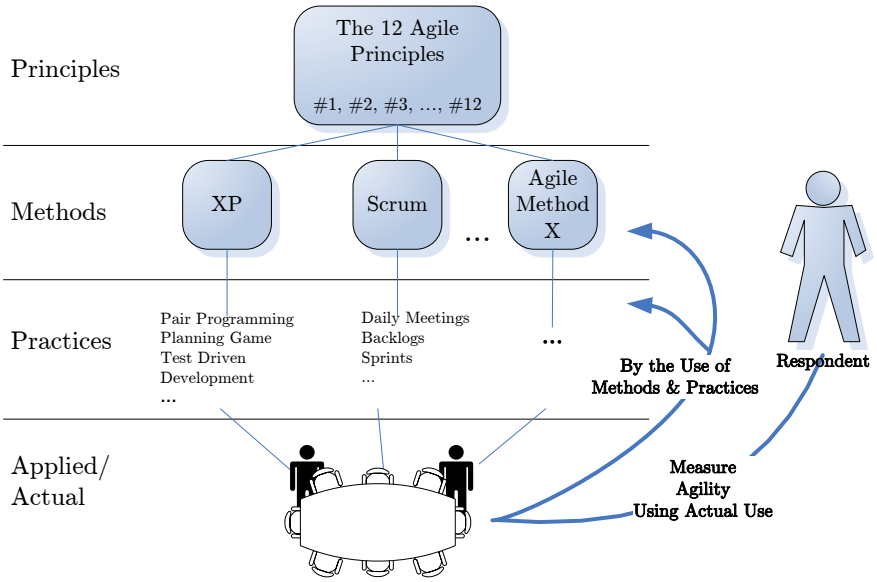


Figure 2.1: Agility is measured by measuring adherence to practices and methods, i.e. what people actually is carrying out, is compared to the descriptions of the practices and methods.

The model in Figure 2.1 extends Figure 1.4 with an *Applied/Actual* layer, which refers to what respondents are actually doing. By capturing what respondents are actually carrying out, and comparing this to the descriptions of the applied methods and practices, the level of conformance between the two can be measured and used as an indicator of agility. XP is rather nice in this aspect, as it describes 12 practices to follow in order to do XP. The *Adherence Metrics* in the XP Evaluation Framework more or less builds on these practices. Some of the metrics, in the XP Evaluation Framework, are objective and some are subjective, which will be mentioned in the following.

Objective Measures

The reliability of objective measures is higher than that of subjective measures as they ignore transient constructs such as emotional states [9]. However, constructing objective metrics in order to measure people's use of certain practices and methods are difficult. Layman et al. [23] examines the *test first* practice in XP and defines a number of metrics for measuring adherence to the practice; *code coverage*, *test run frequency*, *test class to user story ratio*, etc. Significant problems with these measures are that the data is hard to collect and very often requires participation from developers. E.g. gathering *test run frequency* data require that developers make a timestamp for each test-run executed. A possible solution would be to automate the gathering of as many metrics data as possible.

In order for a team to adhere to the *test first* practice, data should not only be collected once or twice. To ensure that tests are not written and executed in phases, but continuously, it is necessary to collect data throughout a project. If the metrics data gathering is not fully automated, this would impose further overhead on developers, as they are burdened with gathering data in addition to all their project related responsibilities. Such enforcement would clearly disrupt the value of the collected data.

There exists tools to collect data for objective metrics such as *code coverage*, however, others would require the development, implementation and use of new tools. Gathering data for measuring the *test run frequency* would likely be possible, whereas an adherence metric such as *test classes to user story ratio* is more difficult to automate.

Given that each practice involves a number of metrics, where the data collection is very difficult to automate and that e.g. XP has 12 practices, the task of fully automating the metrics data gathering would require a very large and cumbersome setup. Many new tools would have to be developed and installed in the environment under study. Also respondents would likely be required to use new tools (e.g. to store user stories along with updates and changes electronically instead of using hard copy index cards). Even if the obstacles of defining metrics and automating the data collection is overcome, there still is a problem with measuring what cannot be measured, e.g. the value of tests. *code coverage*, *test run frequency*, etc. can be useful as they indicate that some level of testing is occurring, however, they are superficial measures. The percentage of code covered by tests does not indicate whether the critical parts of the code are tested or if the tests are meaningful at all. Another example is *pair programming*. How can the flow and interaction of two people doing pair programming be measured objectively? There is a big difference between *putting-your-minds-together* and to sit at a screen where one is typing code and the other reads comic books. In order to meaningfully measure such things, subjective measures such as external reviews or observations are necessary.

Subjective Measures

In the case study carried out by Layman et al. [23], the adherence metrics also include subjective measures. These are gathered through a questionnaire where team members are asked to rate their adherence to each of the practices in XP. E.g. *What percentage of your work (design, analysis, coding) was done in pairs?* [24]. The validity of this data is thus

coupled with the developers' ability to rate themselves fairly and also on their knowledge and understanding of the practices they rate their adherence to. The value of such metrics highly depends on what they are used for. If, e.g. the metrics are used internally in a team for spotting areas of potential problems with the implementation of XP, the metrics should do fine. However, for assessing the agility of a development team or measuring a team's adherence to XP, it is believed that such metrics are quite vulnerable.

It is crucial that developers do not misinterpret or have different views on what each practice actually implies. For instance, it is easy to grasp that the *test first* practice of XP implies that a developer tests before he begins to code. The practice is nice and simple. However, the purpose of the practice assumes that each test written is reasonable and that an adequate number of tests are created. The practice is not followed, if the developer simply writes a couple of "naive" tests just as a means to begin writing code. Such behavior could perhaps be expected by an individual who either disapproves of the practice or do not understand it thoroughly. The *test first* practice (as all the practices of XP) calls for a fundamental belief in and understanding of the practice. If this is not considered when measuring the adherence to a method, such subjective metrics are pointless.

When examining the results in the case study done by Layman et al. [23], a few things are interesting especially concerning this aspect of developers understanding of practices and their ability to rate their adherence to them. One of the more important practices of XP is having an *in-site customer*. Kent Beck explains: *"The point of customer involvement is to reduce wasted effort by putting the people with the needs in direct contact with the people who can fill those needs."* [2].

By working closely with the customer, developers attempt to understand and clarify requirements as the customer can provide quick feedback on the basis of the present system. In the case study, Layman et al. [23] records that the project has 3 customers, all remotely based. One is domestic and two are international of which one is overseas and multiple time zones away. Data for the *on-site customer* metric is gathered by asking each respondent: *"What percentage of the time do you get quick interaction with your customers when needed?"* [23]. Surprisingly the average score to this question is 70% adherence with a 23.6 deviation. Why (or how) does this metric score so high 70% with so remotely placed customers? A possible answer could be that developers, when stating the adherence to the practice, are referring to their interaction with a customer representative and not the real customer recorded by Layman et al. [23] . However, the evaluation framework clearly state how the data for the customer-metric is gathered:

"Record the number of customers (or customer representatives) with which the team interacts. Document if the customer is located on-site, in another city, country, or time zone. Also document if the customer is from a different culture or uses a different language." [24]. If this is the case, the customers must either travel a great deal to meet with developers or other forms of communication must have been used, e.g. phone, e-mail, or instant messaging. Beck, as well as the rest of the Agile Manifesto authors, clearly advocates physical involvement of customers, customer representatives or users during a project. Anything other than face-to-face communication is not regarded as an agile approach. Bottom line is, that there is a large possibility that developers in the case study are interpreting the *on-site customer* practice differently then what, according to Beck, is considered agile.

2.2.2 Room for Interpretation

The difference in developer's knowledge and understanding of agile practices and methods is also recorded in a survey research by Reifer [26]. Reifer uses a questionnaire to identify *What agile practices early adopters use?* and collects data from 28 companies of which 14 claim that they are using agile methods. However, when it comes to the specific practices the teams are using and which they consider agile, the list is long and diverse. Generally the respondents disagreed on what the best agile practices are. In addition, a difference in the actual form of process and how informal or flexible an agile process should be, are also recorded. Reifer's research study shows that respondents agreed that agile development must be a cyclic process and involve builds and increments done concurrently. In addition they agreed that agile projects *"must involve collaborative organizations that include participation by all stakeholders during development"* [26]. However, they disagreed on who the stakeholders actually are and how involved they should be.

Another problem with using adherence to practices or methods to measure agility is illustrated by Poppendieck [11]. She notes the following about principles and practices:

"Principles are underlying truths that don't change over time or space, while practices are the application of principles to a particular situation. Practices can and should differ as you move from one environment to the next, and they also change as a situation evolves." [11].

But what does adherence to a method or practice actually indicate? This could be a reasonable question to ask. According to Poppendieck [11] it does not mean anything per se. Adherence to methods or practices does not indicate conformance to the underlying principles of a methodology. Conversely deviating from a method or practices does not indicate a disagreement with or inability to carry out the underlying principles of a methodology. Thus adherence to methods or practices is questionable as a yardstick for agility. It might be that a team does something a bit different than what an agile practice (*on-site customer*, *pair programming*, etc.) prescribes, and still be considered agile. The agile authors themselves and one of the agile principles mentions that, reflection on what is actually carried out and how things can be optimized is important. Tailoring a method and practices can not necessarily be seen as wrong or an indication of not being agile.

2.2.3 Metrics Gathering Through a Questionnaire

It may be possible to construct objective metrics that can be used to measure some level of conformance to practices and methods. However, doing so through a questionnaire is very difficult. As explained, the gathering of the many objective metrics requires the participation of the subjects investigated and the development, installation and use of tools continuously during a project. This is not possible with a questionnaire. Subjective metrics on the other hand relies on peoples' knowledge of practices, which leaves a lot of room for interpretation. Another problem is to identify and weight the different agile practices to measure adherence to. It is quite likely that using an iterative development approach is more important in order to be agile then it is to do *pair programming*. Even if these obstacles could be overcome, the question still remains whether conformance to methods and practices are a good yardstick for agility.

This implies that using adherence to methods and practices in order to measure people's agility and relying on developer's knowledge and correct understanding of these, is very difficult and problematic through a questionnaire.

2.3 Values and Principles

Values and principles are the core of Agile Software Development, thus the related research is examined with regard to these. Four out of the nine articles and one of the survey reports, employ the Agile Manifesto, its values and 12 principles. However, the agile values and principles are used in different ways in the articles and the survey report. Guntamukkala et al. [21] use the values and principles to classify three different *lifecycle model groups* and place six different software development methods in each group. Lindvall et al. [25] use the agile values and principles to establish a definition of agile methods. Hansson et al. [22], Fruhling and De Vreede [18], and the survey report from Shine Technologies [38] all use the values and principles in order to evaluate agility.

The reviewed articles and the survey report are presented in the following subsections in the order mentioned above.

2.3.1 Classification of Methods

Guntamukkala et al. [21] conducts a questionnaire survey among 74 project managers and software developers, to establish a new alternative way of selecting a software development life cycle model from the perspective of flexibility. Their results show that there are three naturally occurring development model groups; *heavyweight*, *middleweight* and *lightweight*. The last group represents agile methods, e.g. XP, Scrum, etc. They describe the characteristics of lightweight models by referring to the values and principles of the Agile Manifesto and Highsmith [4]. Thus lightweight models are described as supporting the following values:

customer value (focusing on delivering what the customer needs), individual capability (focusing on individual skills and talent), collaboration (focusing on innovation through group interaction), and adaptation (focusing on feedback and harnessing change) and minimalism (focusing on simplicity).

They use these agile values to classify methods like XP, Scrum, etc. into the lightweight models group. In the questionnaire, respondents are presented with a list of methods/models and are asked to report the extent to which they would use a given method/model in specific situations. The results are used to construct a canonical function which assists in selecting a method appropriate for a given set of circumstance (e.g. requirements are constantly changing). However, the assumption underlying this approach is that the respondents are in fact capable of making correct choices regarding which method or model suits different situations the best. This implies that respondents are familiar enough with all the listed methods and models (six in total) and that they are skilled enough to make such choices. Perhaps it would be more appropriate to examine the values and principles underlying each of the three model groups and provide the respondents with of list of their different

properties (customer collaboration, requirements specification, etc.). The respondents are then required to state the extent to which the different properties suits different situations. Such properties would be clearer for respondents and easier to relate to situations than methods and models.

2.3.2 Defining Agile Software Development

Reifer [26] conducts a survey among software companies and recommends that prior to choosing an agile method or agile approach to software development; one should clearly define what "agile methods" mean. The Agile Manifesto was established to describe what agile people should value and which principles they should follow. In an eWorkshop, eighteen agile and methodology experts gathered to discuss and debate Agile Software Development [25]. The article presents the working definition of agile methods that the participants agreed on:

> - *Iterative (Delivers a full system at the very beginning and then changes the functionality of each subsystem with each new release.)*
> - *Incremental (The system as specified in the requirements is partitioned into small subsystems by functionality. New functionality is added with each new release.*
> - *Self-organizing (The team has the autonomy to organize itself to best complete the work items.)*
> - *Emergent (Technology and requirements are "allowed" to emerge through the product development cycle.)*
>
> *All Agile methods follow the four values and 12 principles of the Agile Manifesto.*

Besides referring to the Agile Manifesto, their definition highlights four items which characterizes agile methods. They should be iterative, incremental, and emergent and teams should be self-organizing. The definition is not discussed or argued more thoroughly in the article, as to how and why exactly these four characteristics are more expressive of agile methods than the values and principles in the Agile Manifesto. As a definition it is, however, not more precise than the Manifesto values and principles.

Larman [8] states the following about defining agile methods: *"It is not possible to exactly define agile methods, as specific practices vary ... "*. If it was possible to definitively define agile methods, the Agile Manifesto authors would have created a definition of Agile Software Development, instead of creating a Manifesto and the supporting principles. Creating such a definition is not possible, because the agile methods came before the Agile Manifesto, and the Manifesto is an umbrella that covers the common values and principles within the agile methods. The closest thing to a definition would be the Agile Manifesto, its values and principles.

2.3.3 Evaluating Agility

Fruhling and De Vreede [18] did an action research study on experiences with the implementation of XP in a web-based distributed information system. The system was an emergency response center and the objectives of the research study was to investigate how XP could be applied in a mission critical context, even though theories suggests that agile methods are not suited for this [15] [19]. Adherence to the XP practices are first assessed by comparing them to what they have actually done. The agile values in the Agile Manifesto are examined the same way and the article describes how the project applied to each of the values. E.g. the first valued statement in the manifesto states: *Working Software over Comprehensive documentation*. When addressing this value, Fruhling and De Vreede [18] states:

> *"Because of the reduced amount of documentation, several benefits to the project resulted. ... This streamlined the development process. Decisions were made verbally and the developers acted on them immediately ...".*

They focus a lot on not doing documentation and neglect to mention anything about the *working software* part of the statement. How did they handle working software? What did they mean working software was? Was it an evolutionary prototype? Did the users/customers have any opinion on less documentation and more working software? These questions are not taken into account. The other three values are handled in the same way, where emphasis is put on specific items within each valued statement. Employing the agile values and examining how they apply to their project is a reasonable approach. However, they do not treat the agile values fairly and sometimes neglect to reflect and relate to them with regard to their own experience during their project. It is easy to look at a value and say, this is what the values say and this is what we have done. But this is not enough. One has to examine the values in more detail to see if the value has been understood correctly. Here, the 12 agile principles would be very useful, as they describe how the values are supported and thus provides more detail. So to be able to use the agile values and principles as a mean of measuring agility, one has to examine the values and principles in detail to understand them fully.

Hansson et al. [22] conduct a research study on how agile industrial software development practices are. They interview software developers from five different companies and examine the agility of their actual use. The values in the Agile Manifesto are used to evaluate whether the software processes can be seen as agile and to what extent. They conclude that it all depends on which of the different values are used to judge agility. It also depends on the characteristics, not only of the company, but also of each single project. Digital Focus [31] which is a leader in Agile Software Development and integration services, conducted a survey that states the following with regard to what it entails to be agile: *"Adopting agile is not just a process of understanding and applying different technical practices. It is a mindset shift that must take place within the executive, business and IT communities of that organization."*. This indicates that the researchers, Hansson et al. and Digital Focus have different opinions on what it constitutes to be agile.

Shine Technologies [38] conduct a survey on experiences using Agile methodologies. The questionnaire includes ten questions and 131 respondents completed it. Out the ten ques-

tions, only two employ the agile values. The first question reads: *"What feature of your Agile processes do you like the most?"*. The respondent is given five possible choices; *Respond to change over following a plan*, *Relationships over contracts*, *Code over documentation*, *People over process* and *Other*. However, as these choices are modified and smaller versions of the four valued statements in the manifesto are presented, the reliability can be questioned. The results show that the respondents are most positive towards the feature *"Respond to change over following a plan"*. Interestingly enough, this is the only feature which is not modified significantly. Moreover, if concrete conformance to the manifesto is pursued, it is problematic to alter some of them for readability. E.g. "Relationships over contracts" might not be interpreted in the same way as "Customer collaboration over contract negotiation". The Agile Manifesto statement puts emphasis on the customer, whereas the modified version removes the customer from the sentence. Without careful examination of the valued statement and what it implies this is a problem. It might remove valuable information and thereby introducing a bias as the question is not interpreted the same way by the researcher as it is by respondents. E.g. the researcher couples *"Relationships over contracts"* with the Manifesto statement and the respondent interprets it as creating personal relationships with his team or referring to oral agreements.

The second question they ask is: *"What features of agile processes makes you most uncomfortable"*. The possible choices are variantions of the items on the right side of the Agile Manifesto; *"Lack of authority"*, *"Lack of project structure"*, *"Lack of planning"*, *"Low documentation"*, and *"Other"*. Most of the respondents selects *Other*, which could indicate that the question lacks possibility of more questions. However, the question is interesting as it address the possible barriers to Agile Software Development.

The two questions generally lack more possible choice for the respondents to select. For instance, in the first question it is not possible to state if none of the agile features are liked. In the second question the same lack of extra option is present, as it is not possible to state that no agile feature makes them uncomfortable. This limits the use and interpretation of the results, in our opinion, and thus the validity and reliability of their findings.

2.3.4 Actual Use Versus Attitude

When using values and principles to measure agility in a questionnaire, there are two ways to measure agility as described below and illustrated in Figure 2.2:

1. By asking the respondents about their *actual use* and comparing this to the agile values and principles.

2. By asking the respondents about their *attitude* towards agile values and principles.

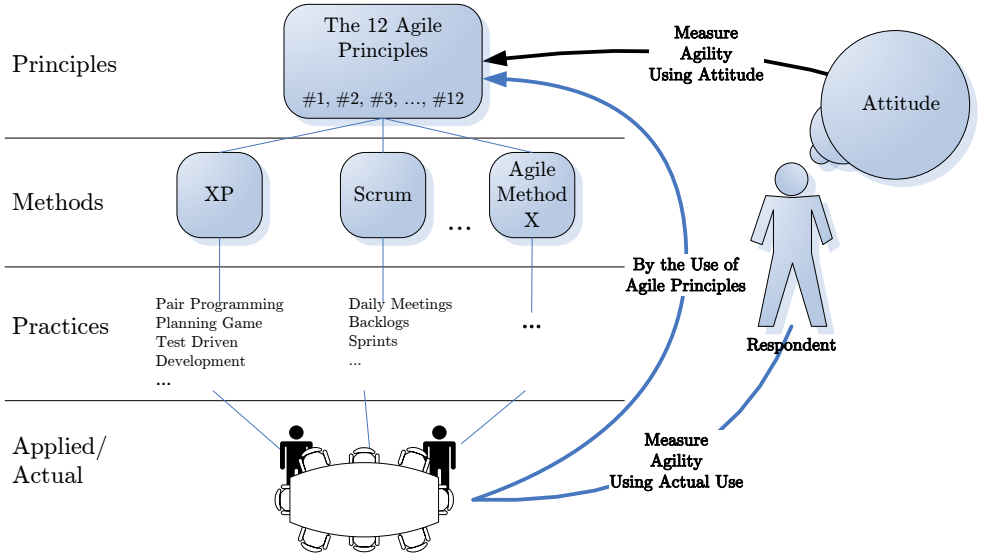


Figure 2.2: The two ways to measure agility using values and principles; actual use or attitude.

Figure 2.2 is a continuation of Figure 1.4 in Section 2.2, but instead of using methods and practices to measure agility the values and principles are used. The first way to use values and principles is by looking at the respondents' actual use of them. This entails asking questions about how the respondents work and thereby gaining an overview of the phases and tasks involved in their work. The most appropriate data collection methods for this kind of research would be observation, in that it enables the researcher to collect the needed information without the subjective interpretations of the respondents. However, the data is still collected through the researchers, and thus subjective. If this approach is used with a questionnaire as the data collection tool, many of the questions would need to be open-ended, in order to cover the choices sufficiently and thereby avoid the "lack of choice" problems in the Shine Technologies survey [38]. However, open-ended question negatively impacts the response rate, as respondents are burdened with formulating their own answers. So the approach of looking at actual use and comparing this to values and principles may not be the best choice for a questionnaire survey.

Ruling out using actual use and comparing it to values and principles, one option is left and that is to ask the respondents about their attitude towards Agile Software Development values and principles. But how can attitude be used to measure agility?

Merriam-Webster [35] defines attitude as:

1. *A mental position with regard to a fact or state <a helpful attitude>.*
2. *A feeling or emotion toward a fact or state.*

In others words, it is a specific state of mind towards a fact or state. This is also how Digital Focus [31] interprets "agility": *"Adopting agile is not just a process of understanding and applying different technical practices. It is a mindset shift that must take place within the*

executive, business and IT communities of that organization.". This indicates that agility is as much a state of mind as it is a set of technical practices. So attitude is ideal for measuring respondents mindset and thereby their agility. Also researchers within in questionnaire theory have developed an extensive set of tools for measuring people's attitude.

These include various attitudinal scales such as Likert and Thurstone scales, which enables the collection of respondents attitude towards agile values and principles.

2.4 Project Context Factors

Some of the related research articles investigate the context of a specific software project. The reason for this is that project context matters when discussing agility. Some articles claim that certain context factors can divide projects into agile or plan-based project home grounds [23] [18], while others look at context factors in a more exploratory fashion [21] [31] [25]. Nevertheless all agree that project context is very important when choosing a software development methodology. This is not only a point concluded in these articles but also made by Cockburn, the author of the Crystal family of agile methods, who states that project context is all important when choosing the development method [8]. Some articles also use context features as a way of understanding the generality and utility of the findings [23]. The articles, however, do not entirely agree upon what context factors are the most important and the extent to which they influence what development method to use.

2.4.1 Context Factors in the XP-EF

Layman et al. [23] uses the XP-Evaluation Framework (XP-EF) to evaluate an XP project. The framework is composed of three parts, of which one concentrates on project context. It takes a broad approach defining six contextual categories of interest.

1. Developmental factors (Size, criticality, dynamism, personnel, and culture; proposed by Boehm and Turner [15])
2. Sociological factors (e.g. team size, education level, experience level, etc.)
3. Project specific factors (e.g. domain, person months, nature of project, etc.)
4. Technological factors (e.g. soft. dev. meth., project mgmt., language, etc.)
5. Ergonomic factors (e.g. physical layout, distraction of office space, customer communication)
6. Geographical factors (e.g. team location, customer cardinality and location, supplier cardinality)

All factors are explained and discussed in detail in Layman et al. [24]. The article tries to objectively collect all these context factors to know as much about the project and its surroundings as possible. The collection method for all these context factors varies from

observation to the use of CASE tool project statistics. The primary goal of collecting these context factors is to get a foundation for comparison among other projects and thereby learn more about how a specific method works and performs. However, the article also states that this level of rigor in the collection of context factors is very difficult and time consuming, coupled with the fact that some measurements, like number of New/Changed classes, are impossible to collect without developer participation or tool support. Furthermore as these characteristics are not used as specific indicators for agile or plan-driven suitability, they are not all relevant for the purposes of our survey.

2.4.2 Context Factors Proposed by Glass

Hansson et al. [22] use four context factors proposed by Glass [19] to evaluate five Swedish software companies. The four context factors are as follows:

- Application Domain
- Project Size
- Criticality
- Innovativeness

Application domain looks at the kind of software that is developed. This is typically based on the business type or domain the software is developed for and is subjectively evaluated. In the case of Hansson et al. [22], the researchers categorized each of the five companies instead of individual projects, which is difficult if the company in question has a diverse product line. This also means that it is difficult to create a generic list of application domains that is applicable to all projects.

The second context factor is project size. Hansson et al. [22] uses the number of developers on a given project as the measure for project size. This is something that can be measured objectively, which greatly benefits the accuracy of the measurements on this particular context factor.

The third context factor that has been believed to correlate with the use of agile methods is software criticality. Many believe that software developed to meet high criticality standards cannot be built using agile methods because of their lack of oversight, planning and documentation [15] [22]. However, agile authors and practitioners have argued that this is not the case if sufficient steps are taken to ensure software criticality within the agile method. Here they refer to the emphasis on test driven development in XP [25]. Hansson et al. citeIndustrialSoftware investigates the correlation between Agile Software Development practices and software criticality. To categorize the software criticality they divide criticality into three categories, high, medium and low. Their results indicate that the most agile companies indeed are the ones that develop low criticality software. However, since none of the companies fall within the category, ”high criticality” and only five companies was selected the results are inconclusive and cannot be generalized to the entire field of software development. There seems, however, to be a correlation between Agile Software

Development practices and software criticality. Also the use of a classification scale ranging from high to low is subject to personal interpretation. This means that if used in our survey respondents could interpret the scale differently and thereby influence the result.

The last context factor by Glass [19] is innovativeness. Hansson et al. [22] takes the term innovativeness to mean innovative use of technology and not innovation of technology. This context factor is again evaluated on a scale of low, medium, and high and none of the five companies fall within the high category. It is an interesting context factor in that, the use of agile methods is often associated with innovative projects. This stems from the belief that innovative projects are in a constant state of chance and therefore not suitable for plan-driven development. Taking this viewpoint makes innovativeness an appealing context factor within the scope of our survey. However, again the subjective evaluation of the scale used introduces problems with interpretation and evaluation when used in a questionnaire.

2.4.3 Agile and Plan-Driven Home Grounds

Layman et al. [23] and Fruhling and Vreede [18] both use a framework developed by Barry Boehm and Richard Turner [15] to categorize projects as agile or plan-driven using a number of context factors. These context factors have been selected for their ability to divide agile and plan-driven methodologies. Both articles use this framework to indicate if the project under examination is best suited for an agile or a plan-driven approach. Boehm and Turner present five context factors that define their framework:

1. Personnel: The skill level of the individual developers within the team
2. Dynamism: The number of requirements changed per month
3. Culture: The number of people thriving on chaos versus order
4. Size: The number of people on the team
5. Criticality: Criticality of the software in the development project

These five context factors can be plotted on a polar chart as illustrated in Figure 4.8. Boehm and Turner [15] argue that these five context factors divide the suitability of a project into agile and plan-driven domains or home grounds as they call it.

The Boehm and Turner context factors can be divided into two groups, which distinguish themselves from each other in that one can be objectively measured while the other is measured with some degree of subjectivity. The objective group is comprised of the *Criticality*, *Size* and *Dynamism* factors as they have clearly defined scales of measurement. This makes these context factors easy to investigate quantitatively via a questionnaire. *Personnel* and *Culture* on the other hand has to be evaluated subjectively, which always has the possibility of varying depending on each individual doing the evaluation. Nevertheless the five context factors are all indicators of agile or plan-driven suitability, where criticality, size and to some extent dynamism are already adopted by Glass [19].

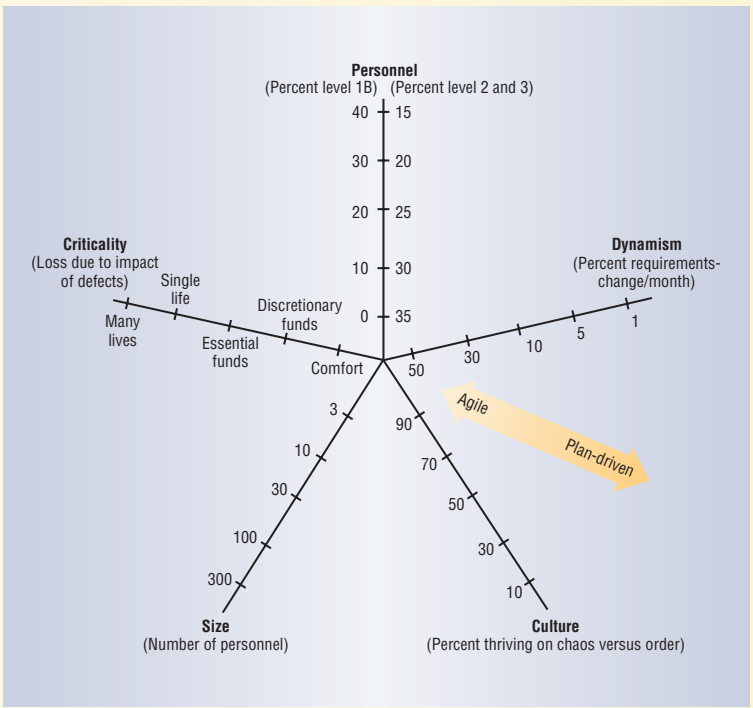


Figure 2.3: Polar chart displaying the five context factors developed by Boehm and Turner [15].

2.5 Related Research Findings

The articles' handling of agile software development differ in three main areas. These areas are method and practices, values and principles, and project context. The first two tackle the problem of defining a standard on, which agility can be measured, and the third looks at potentially agile project context factors. These are the aspects that are of interest to us in this project in order to answer our the research questions.

Measuring Agility

Some articles use the assumption that the use of an agile method or a set of agile practices makes you agile. This is done without looking into the degree to which the method is used or into the development teams understanding and proper use of the practices. Because of this assumption one has to delve deeper into each team's use of an agile method to obtain sufficient assurance that they can in fact be categorized as agile. However, none of the questionnaire surveys have done anything beyond asking about specific methods and practices because an in depth inquiry into use and understanding of a method or practices is not feasible through a questionnaire. Also the differences between the many agile methods and practices mean that they should all be included, to get an accurate estimate of the

overall agility. This in turn leads to a massive and comprehensive questionnaire because not merely XP or Scrum can be used to measure agility. This leads to the conclusion that adherence to methods and practices alone cannot determine agility in our survey.

Other researchers have tackled the problem of agility by using the Agile Manifesto's valued statements and principles. This approach is used in an attempt to straddle all the values and principles that are deemed agile, without having to limit themselves to a specific method or practice. This approach has significant advantages because it goes beyond methods and practices when looking at agility and tries to se agility as a set of values. However, the Manifesto's valued statements are somewhat vague and philosophical, and leave plenty of room for interpretation. This means that they are ill suited for a questionnaire. The principles, while being more concrete than the valued statements, still leave room for interpretation and the thoughts and intentions behind them are not always clear. The articles using the Manifesto's valued statements and principles have not tried to alleviate these shortcomings and merely used them as is. This approach may be possible when the data collection is done through a case study and the judges of agility are the researchers themselves, but could prove erroneous when conducting a survey. For this reason further research on operationalizing the agile values and principles should be done before using them in our survey. This is done in Chapter 3.

Furthermore there are two ways of looking at values and principles. One is through the respondents' actual use and through the respondents' attitude towards these values and principles. Given that the actual use approach conflict with the questionnaire data collection method and that agility has been tied with attitude, the measure of agility should be done using attitude.

Agile Project Context Factors

The articles looking at project context have done this for a variety of reasons. The main reason we are interested in is the identification of project context factors that have a correlation with agility. Both Glass [19], and Boehm and Turner [15] have highlighted context factors they think should be taken into account before implementing either an agile or plan-driven approach. At first glance they seem to have only project size and criticality in common, but if Glass' innovativeness factor is seen as described in section 2.4.2, it resembles Boehm and Turner's dynamism factor a great deal. With this in mind the Boehm and Turner factors are somewhat more clearly defined and less subject to interpretation and therefore more suitable in a survey. For these reasons the project context factors used in our survey are the ones proposed by Boehm and Turner [15].

3 Operationalization of Agility

The first step towards measuring the agility of IT professionals is the operationalization of Agile Software Development into measurable areas. This chapter first introduces the literature review of the agile manifesto authors. Then the findings are presented in terms of an examination of each of the 12 principles. Finally 7 areas, where agility can be measured are identified.

3.1 Literature Review of the Agile Manifesto Authors

In Section 2.3 it was identified how the values and principles are open for interpretation if used by themselves. The unclarity of the values and principles therefore makes further examination necessary. However, given the huge amount of information on the subject a structured approach is necessary. This involves a careful selection of information sources and setting up some form of theoretical framework in order to conduct a literature review as explained by Kumar [6].

The obvious source to use in the review is the authors who wrote the Agile Manifesto. The 12 Agile principles provides more detail than the four valued statements and would therefore be excellent as a framework for the review.

3.1.1 Selecting Authors

Reviewing all the information from the 17 manifesto authors would be too much. However, a handful of the authors are very dominant compared to the other authors, when it comes to publishing books and articles about Agile Software Development. In most circumstances selecting those who "speak loudest" is not considered the best approach for sorting valid sources from non-valid ones. A better approach would be to conduct a citation analysis,

where reference to agile authors are counted [6]. A *manifesto author sample* of 3-4 authors should be sufficient as input for the review.

Those manifesto authors who are cited often in related literature are assumed to be good sources. When other researchers cite a manifesto author, the author is presumably reliable and understandable and the specific work cited (article, book, etc.) would be fairly accessible.

In this project the citation analysis is done in three ways.

- Citation Analysis of agile manifesto authors using Web of Science
- References to agile manifesto authors found in highly cited Agile Articles
- References to agile authors found in Textbooks

The first uses the WOS citation analysis feature and concerns only published articles written by agile manifesto authors. The second concerns a manual review of reference made to the Manifesto authors in highly cited articles which concerns Agile Software Development. The third involves manually reviewing references made to the agile manifesto authors in software development textbooks.

Citation Analysis of Agile Authors using Web of Science

Using this WOS *citation analysis tool* it is possible to calculate the total number of reference made to articles that are written by each of the agile manifesto authors and published on WOS. Using the tool it is easy to 1) distinguish authors with few published articles from authors with many publications and 2) distinguish heavily cited authors from lightly cited ones. The result is that 8 out of the 17 manifesto authors have published articles on the journal database about Agile Software Development. Three of these authors are cited by others; Alistair Cockburn, Jim Highsmith and Ron Jeffries.

It is believed that the poor result reflects that a small portion of the articles written by the manifesto authors about Agile Software Development are published on the WOS journal database.

3.1.2 References to Agile authors in Agile Articles

The second approach is to use the WOS journal database to find those manifesto authors who are referenced in the most cited articles about Agile Software Development. The 10 most cited articles about Agile Software Development found on WOS is reviewed and references made to manifesto authors are counted. Out of the 10 articles examined, Kent Beck and Alistair Cockburn is referenced 4 times each, i.e. by 4 articles out of the 10. Jim Highsmith is referenced 2 times, Martin Fowler 3 times and Robert C. Martin 2 times.

3.1.3 References to Agile authors in Textbooks

Two textbooks are used in this review of references; Roger S. Pressman's *Software Engineering - A Practitioner's Approach* [12] and Craig Larman's *Agile & Iterative Development - A Manager's Guide* [8]. The first book is concerned with software engineering processes in general and thus provides a brief overview of Agile Software Development, whereas the second concentrates entirely on iterative and agile methods.

In *Software Engineering - A Practitioner's Approach*, Robert S. Pressman [12], refers to books and articles by Jim Highsmith, Alistair Cockburn, Martin Fowler, Ken Schwaber and Kent Beck when describing what Agile Software Development is.

In *Agile & Iterative Development - A Manager's Guide*, by Craigh Larman [8], three books are recommended for further reading on Agile Software Development. Two of these are written by manifesto authors; *Agile Software Development*, by Alistair Cockburn [3] and *Agile Software Development Ecosystems* [4], by Jim Highsmith.

3.1.4 Sources for the Review

After the citation analysis and examination of references to agile authors in different articles and textbooks, books and articles of the following four agile authors is used in the review of the Agile Manifesto and principles:

- Jim Highmisth
- Alistair Cockburn
- Kent Beck
- Martin Fowler

Jim Highsmith and Alistair Cockburn, clearly distinguished themselves from the rest of the manifesto authors as their names are prevalent in the articles and textbooks. In addition, their books on Agile Software Development, *Agile Software Development Ecosystems* and *Agile Software Development* are recommended by Larman [8]. Further more Pressman [12] also refers to both Jim Highsmith and Alistair Cockburn.

Kent Beck and Martin Fowler were also mentioned and referenced by Pressman [12] and in several of the agile articles. Kent Beck is the creator of XP together with Ward Cunningham, and Ron Jeffries.

Martin Fowler co-authored Kent Becks first book on XP. He also wrote an essay about the agile software movement called *The New Methodology* [34]. After Fowler had conversations with other people sharing similar ideas towards software development, the essay got updated and now explores the similarities and differences between the various agile approaches.

The three books written by Highsmith, Cockburn and Beck and the essay by Fowler are the prime source for the review of Agile Software Development.

3.1.5 Review Process

As described by Kumar [6] a literature review involves setting up a theoretical framework and continuously maintaining it while reading the literature. Through this process, subjects and themes are categorized in the framework and in the case of encountering new themes, these are added to the existing framework. The review is conducted by typing the 12 principles into the top column of a 12 columned table. An item from the literature pool, e.g. *Extreme Programming Explained: Embrace Change*, is then selected for review. Upon encountering statements which relates to and clarifies one of the principles in the table, the specific statement is entered into the column of that principle. The results of the review are presented next, in Section 3.2.

3.2 The 12 Principles

The objectives of the operationalization is to pinpoint those areas within Agile Software Development where agility can be measured through a questionnaire. Each of these areas must therefore be measurable and together they must cover the main aspects of Agile Software Development.

In this examination each principle is discussed and clarified based on the review of the four notable agile authors. In addition the agile values that the principles support are identified. This relation is illustrated graphically for each principle where the symbols seen in Figure 3.1 are used.

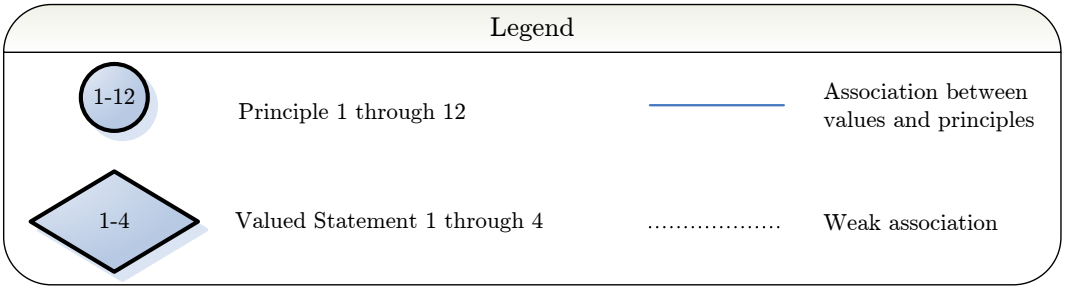


Figure 3.1: Circles are used to describe principles where the number inside the circle refers to the principles number (e.g. principle 1). Diamonds describes a valued statement and again the number inside the diamond refers to the specific valued statement. A blue connection between a value and a principle illustrates that the principle supports the valued statement.

3.2.1 Principle 1

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

One of the things that quickly becomes evident in the review is the high emphasis put on the customer. More than one principle and Manifesto value address this. However, the first principle stands out as it includes the term *"highest priority"*. This wording elevates the importance of the principle. It thereby places

customer satisfaction and *iterative development* at the very top of what Agile Software Development is. The importance of satisfying the customer with valuable software is also described by Jim Highsmith and Martin Fowler. Highsmith states:

> *In the final analysis, the critical success factor for any method - Agile or otherwise – remains whether or not it helps deliver customer value.* [4]

Highsmith presents his view on what the most important thing in software development is, which is to deliver customer value, regardless of the approach. Fowler elaborates further on how agile and traditional software development differs in this aspect:

> *"(In the traditional view) A project that's on-time and on-cost is considered to be a success. This measurement is nonsense to an agile environment. For agilists the question is business value - did the customer get software that's more valuable to them than the cost put into it."* [34]

Here Fowler presents two different views on project success. The first view, that projects which are on time and within budget are de facto a success, is considered questionable. Instead a project and its success, should be measured in customer satisfaction. It might seem trivial to state this. However, the problem and the point that Fowler is making becomes more apparent, when e.g. looking at how a software consultant company such as the Standish Group often cited in literature and the software community, defines project success. They operate with three levels of project outcomes; *Success*, *Challenged* or *Canceled*. Challenged projects are projects which are over-time, over budget or missing functionality. In Fowler's (and agilists) view this *"is nonsense"*. Even a canceled project can be considered a success. E.g. if canceled early in the project the customer can save money and move forward. The success stems from the customers level of satisfaction, not conformance to a plan or budget.

This heavy emphasis on delivering business value in order to achieve success, is the reason why Agile Software Development is concerned with customer collaboration and why iterative development is important. At every iteration the customer decides what is valuable. Highsmith explains the process:

> *"Every iteration, the customer gets to change priorities based on features delivered, features wanted next, and changes requested from previous iterations. The development team's responsibility is to inform the customers what the impact on cost and schedule will be and to present alternatives that might be faster or lower cost, but in the end, the customers are in control."* [4]

For some developers this level of customer involvement might mean a big shift in responsibility. In the agile view developers are viewed as technical consultants. The final word on all business decisions (e.g. the specific placement of a login-panel) belongs to the customer or someone with comprehensive business knowledge and not the developers.

The Agile Manifesto values which this principle supports are illustrated in Figure 3.2.

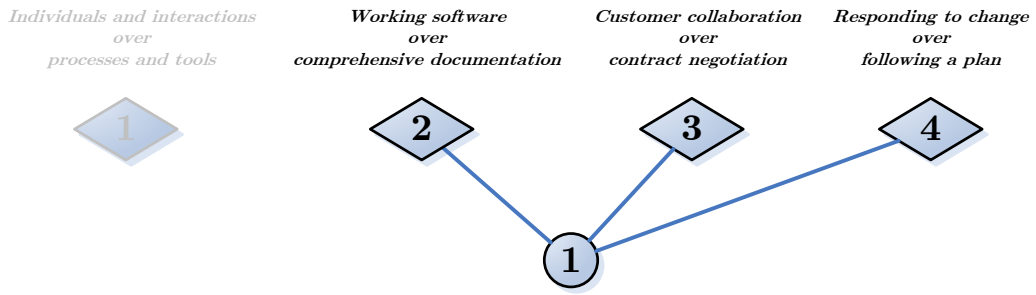


Figure 3.2: The values which principle one supports.

First of, principle one supports the second valued statement. The statement indicates that *"working software"* is valued more than *"comprehensive documentation"*, and by following the first principle's idea of satisfying the customer with valuable software, a clear emphasis on software as opposed to documentation is achieved.

In Figure 3.2 it is also illustrated that the principle supports the third valued statement in the Agile Manifesto. As mentioned above, the principle places delivering something valuable to the customer at the top of Agile Software Development and this emphasis on the customer and his needs are also expressed in the third valued statement. Instead of writing a fixed set of requirements (a contract) from the beginning, the customer is "made a part of the team" (collaboration). This way the customer is in control of the project and can steer it towards success as he defines it.

The principle also has a strong link to iterative development, as it places *"early and continuous delivery of valuable software"* as a top priority. Iterative development supports the fourth and last of the valued statements. The theme here is "change". Although the valued statement does not specify a specific type of change, it indeed concerns changing requirements. By dividing the software process into increments and only freeze the current working increment, requirement changes can be applied to later increments.

3.2.2 Principle 2

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

It is clear from the beginning that this principle is tied strongly to the fourth valued statement, *"Responding to change over following a plan"*. This is illustrated in Figure 3.3.

The principle clarifies three things about the valued statement; 1) It describes the reason behind it (it adds business value); 2) It identifies a specific situation where the value must be supported (when requirements change); 3) It specifies a certain attitude and behavior towards change (welcome change).

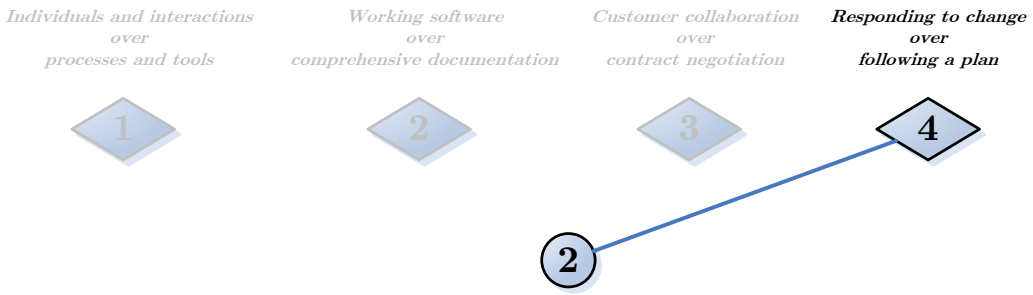


Figure 3.3: The values which principle two supports.

Fowler elaborates on requirements changes and their relation to the customer and business value:

> *"Often the most valuable features aren't at all obvious until the customer have had a chance to play with the software. In today's economy the fundamental business forces are changing the value of software features too rapidly. Even if the customers can fix their requirements, the business world isn't going to stop for them."*[34]

Most software practitioners agree with Fowler's view on valuable features and how these change rapidly. Pressman specifically states that no one is against agility, i.e. the ability to adapt to the changing requirements, however, the question is how to best achieve it [12]. Cockburn provides an agilist's answer:

> *"Agile processes can take on late-changing requirements exactly because of early and frequent delivery of running software, use of iterative and timeboxing techniques, continual attention to architecture, and willingness to update the design."*[3]

Cockburn mentions several agile techniques which accommodate adaption to change. It is clear that iterative development, the use of timeboxing and early and frequent delivery of working software plays an important part. The idea of *continual attention to technical excellence* is examined in the ninth principle. Other agile authors and the value supported by principle two also stress the importance of the attitude towards change. Highsmith argues for a need to view changes as the result of ongoing improvement in information instead of viewing changes as the result of errors (bad planning).

3.2.3 Principle 3

3) Deliver working software frequently, from a couple of weeks to a couple of months, with a reference to the shorter timescale.

Larman [8] notes that iterative development is something which *"lies at the heart of agile methods"*. This process of delivering software in small increments is an important part of Agile Software Development and is exactly what the third principle is concerned with. The principle addresses the timescale or size of an iteration

(a couple of weeks to a couple of months) and it also states, that delivered increments must be *"working software"*.

The review shows that the specific size of an iteration is something that the authors themselves argues about. Highsmith states that three to six weeks is a suitable size. Cockburn recommends iterations of one to three months cycles, but acknowledge that even 4 months cycles are perfectly possible. Kent Beck is most extreme on this matter. He suggest that iterations should be no more than a single week. The reason for this dispute is pointed out by Cockburn:

> *"The duration used for deliveries needs to be negotiated on a project-by-project basis, because delivering updates on a daily or weekly basis can cause more disturbance to the users than it is worth."*[3]

What Cockburn refers to here is that the people with the business knowledge, be that customers or users of the final system, might not have time or be able to use and provide feedback on a new system every week. However, Highsmith draws a distinction between "delivery" and "release". He argues about valid reasons for business people not putting code into production and states:

> *"Still, that shouldn't stop a rapid cycle of internal deliveries that allows everyone to see what's happening and enables constant learning from a growing product."* [4]

Thus it may very well be that a customer is unable or does not want to receive a release every week or month, but this should not affect the iteration size, e.g. increasing it to 2 months.

In spite of the different viewpoints on iteration size, iterative development play a very important part in Agile Software Development, as many activities are build into the iterative process. Activities such as *customer reviews and input*, *team meetings* and *testing* can not take place unless iterative development, to some extent is being used. The specific view on the size of iterations does divide people between being agile or not. It is clearly believed, that iteration size is a matter of the specific project and its context. The important parts to focus on when addressing iterative development is that the before mentioned techniques and properties, timeboxing and small increments are present, and that each increment delivered this way is a subset of the final system in that it delivers the expected business value.

The agile values of the Manifesto which this principle supports are illustrated in Figure 3.4.

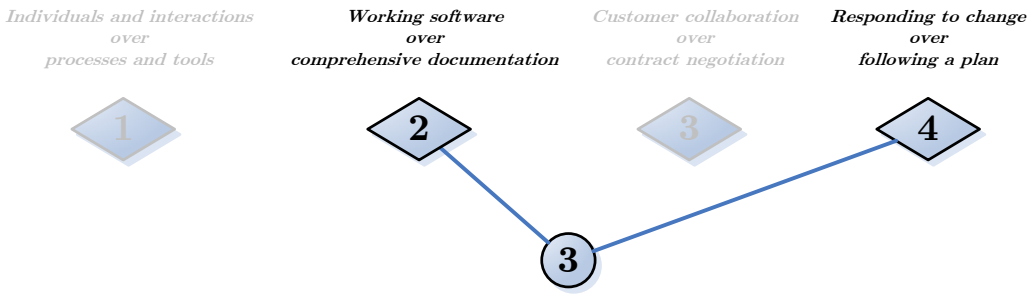


Figure 3.4: The values which principle three supports.

As the figure illustrates, two values are supported by the principle. The first value concerns working software. The principle clearly argues that working software should be delivered and not documentation. This part is discussed more thoroughly in the review of principle 7. The third principle also supports the fourth valued statement. The point of delivering software iteratively is precisely that a development team is able to respond quickly to changes.

3.2.4 Principle 4

4) Business people and developers must work together daily throughout the project.

In the review of principle one, it is examined how the customer plays an important part in Agile Software Development. This is partly because the customer is viewed as an expert in the business domain. Developers must work closely with the customer not only to ensure that the customer gets the system payed for, but also to provide developers with valuable business knowledge. The fourth principle concerns the relationship between developers and business people. It introduces a requirement, saying that business people and developers must work together daily throughout the entire project, which might seem a bit extreme.

Fowler states the following about developers and the business needs:

> *"They need guidance on the business needs. This leads to another important aspect of adaptive processes: they need very close contact with business expertise."* [34].

Highsmith agrees with Fowler and provides insight to the reasons behind the principle. Highsmith refers to the arrival of new business models such as eCommerce and eBusiness and explains how these new and often complex business processes make it difficult for developers to bridge the domain knowledge gap. They need access to business knowledge. Cockburn uses empirical evidence when arguing for the need for business experts. When addressing principle four directly, Cockburn refers to a study showing a strong correlation between projects' success and links to users. He writes that: *"The best links are through on-site business expertise and daily discussions, which is what the statement calls for."*. [3]

Beck [2] use a notion of the *"Whole team"*. The notion includes people with all the skills and perspectives necessary for the project to succeed. This especially includes the customer and Beck states:

> *"They are who you are trying to please. No customer at all, or a "proxy" for a real customer, leads to waste as you develop features that aren't used, specify tests that don't reflect the real acceptance criteria, and lose the chance to build real relationships between the people with the most diverse perspectives of the project."* [2]

Beck clearly advocates for real customer involvement, anything else removes business value. It would seem that the authors disagree on the importance of the actual customer. Beck arguing for the customer and the other authors being more liberal on the subject. Highsmith admits that although the best thing is to have users available when needed this is not always possible and hence business and systems analysts fulfill a vital role on a team as they also possess valuable domain knowledge.

Principle four imposes a strict view on the frequency of interaction as it uses the terms "must" and "daily". Highsmith explains:

> *"There was debate among the Manifesto authors over whether to substitute frequently for daily, and I think we kept the right word. Of course absolute adherence to a daily schedule may not always work. Working together daily may mean for a couple of hours each morning rather than all day, and it may mean skipping days now and then. However, when daily begins slipping to every other day, and then to once a week, results will suffer accordingly, particularly if time to market is a driver."*[4]

Thus "daily" is not meant as a strict requirement to be upheld what ever the cost and situation. It is something important to strive for. Cockburn notes that *"the statement indicates that the longer the time to get information to and from the developers, the more damage to the project."*. Fowler states:

> *"Agile teams cannot exist with occasional communication . They need continuous access to business expertise. Furthermore this access is not something that is handled at a management level, it is something that is present for every developer. Since developers are capable professionals in their own discipline, they need to be able to work as equals with other professionals in other disciplines."*[34]

The agile values of the Manifesto which this principle supports are illustrated in Figure 3.5.

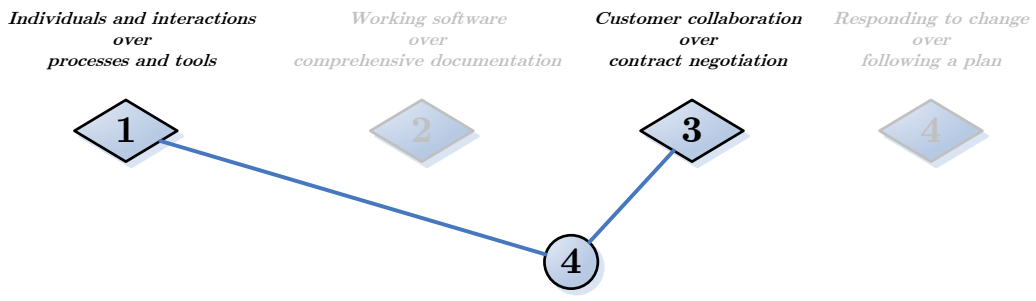


Figure 3.5: The values which principle four supports.

The principle about daily collaboration does not say that developers and business people (customers, feature analyst etc.) must attend at meetings together every day throughout the project. It states the importance of conveying business knowledge to developers and stresses easy (daily) access to this knowledge. As the Manifesto values indicate this involve interaction and collaboration among the team members, business people and customer.

3.2.5 Principle 5

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

The principle highlights people as a very important resource in a software development project. It states that individuals form projects and not the other way around and introduces a management approach based on the concept of freedom with responsibility.

Highsmith and Cockburn are the two of the agile authors who addresses people factors the most. They clearly state that *"individuals make projects work"*. Cockburn explains how he in his early days as an employee, were given the task to find out which development method were most successful and ended up with realizing that methods did not matter. What did matter where the people on the project. God people would always make the project succeed regardless of the method used. However some methods would make it easier for them and some would make it harder. [3]

Cockburn describes that the downside of this insight is that *people are never linear* [3]. Their behavior is very complex and effected by many different factors. Some of these factors are addressed in the fifth principle, namely the *working environment*, *support* and *trust*.

The *working environment* is a theme which Cockburn is especially concerned with. He examines the working environment of the agile team, with focus on how it influences and is able to support collaboration and communication between the team members (developers and business people). Some of the initiatives discussed and suggested by Cockburn are the layout of the room, which should allow face-to-face communication and the concept of *information radiators* for easy and effective distribution of information. The principle also mentions that the team should be given the "support" they need. This refers to things such as education, training, staff interviews, etc. However, *information radiators*, *staff interviews*, *.net certification*, etc. are specific practical solutions, where the principle merely supports the view that attention to the people on a project and their needs are important.

This kind of investment in the development team, should then be followed up with trust, as noted in the principle. Highsmith pictures two different attitudes that the management can have towards developers, a non-agile and an agile:

> Non-Agile: *"I don't trust you to get the job done correctly; therefore, I have to constantly follow up, keep the pressure on or you will slack off, and micro-measure your performance."* [4]

> Agile: *"I trust you to get the job done correctly, and in order to assist in get-*

> *ting the work done, we have to interact in order to monitor expectations and performance."* [4]

According to Highsmith there is no middle ground. Managers either fundamentally trust people or they do not [4]. This trust is highly coupled with decision making. Who (e.g. management or developers) makes what kind of decisions (e.g. technical or business)?

Fowler state that developers must be able to make all technical decisions and refers to the *planning game* in XP as a good example on how estimation is the responsibility of developers. Highsmith is a bit more carefull and mentions that: *"who makes the decision is less important than getting the right "whos" involved in the decision."* [4]. However Highsmith and Fowler do agree on the following:

> *"Decisions must be made by the people who know the most about the situation. For managers, this means trusting in your staff to make the decisions about the things they're paid to know about."* [4]

Though the agile view on decision making argues a shift of responsibility from management to developers, the sharing of responsibility must be equal, according to Fowler. *"Management still plays a role, but recognizes the expertise of developers.*[34].

It is clear that the fifth principle supports the first of the valued statements and this is shown in Figure 3.6.

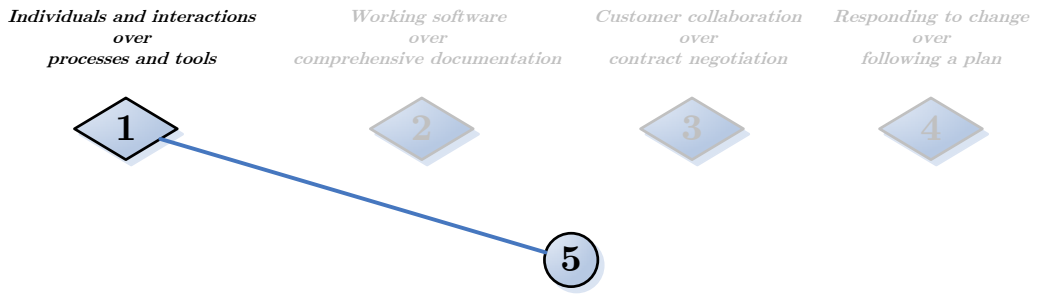


Figure 3.6: The values which principle five supports.

It is important to notice that principle five not only addresses developers, but the agile team. This also includes the customer and therefore the principle touches upon the third valued statement, regarding customer collaboration. However, as other principles address the customer involvement more directly, it is not illustrated here. Instead the principle is seen as being directed more towards providing teams with freedom instead of controlling them.

3.2.6 Principle 6

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

Knowledge transferring takes place all the time, in every day life and in the work place. When someone waves to another person or hands a drawing, they are transferring knowledge. It is an essential part of our lives. The sixth principle concerns how knowledge is transferred to and within a development team and declares that the best approach is face-to-face. Hereby Agile Software Development de-emphasizes other kinds of communication and ways to transfer knowledge, such as different kinds of documentation, phones, electronic chat, etc.

How teams communicate on agile software projects is handled most thoroughly by Cockburn and Highsmith. The problem with many communication mediums such as documents, video, audio, etc. is that it only communicates one way. Questions and answers are not possible when e.g. watching a video or reading a document. The devaluation of documentation which agile methods and principles uphold is often criticized by opponents. They see documentation as a very important artifact in software development and the discipline of producing it must be paid attention. Highsmith elaborates on the dispute:

> *"It's enough to make one scream: "The issue is not documentation, the issue is understanding!" Do the developers understand what the customers want? Do the customers understand what the developers are building? Do testers understand what the developers intended to build? Do software maintainers understand what the developers built? Do the users understand what the system does for them? "*[4]

What Highsmiths pinpoint here is, that the end goal for all kinds of communication, regardless of its shape or form, should be *understanding*. All authors seem to share this attitude towards the documentation versus face-to-face communication discussion. The point seems to be, that while documentation has certain benefits and can indeed be useful, it is not enough. Cockburn notes that the keywords when using documentation should be "just enough" and "barely sufficient" [3]. However Cockburn notes that the cost and quality of face-to-face communication move negatively when the size of a project increases. In these cases documentation (e.g. *information radiators*) serves a "barely sufficient" purpose.

The principle can be viewed as supporting the first of the valued statements. This is illustrated in Figure 3.7.

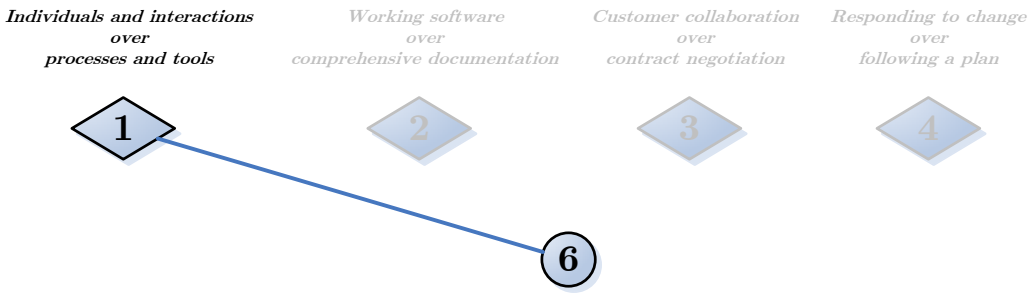


Figure 3.7: The values which principle six supports.

It should be noticed that *"documentation"* is not valued *"individuals and interactions"*,

but *"processes and tools"*. However, having examined the sixth principle it seems that the concept of *"tools"* is wide enough to include e.g. documentation tools, e-mail, chatting-programs, etc. Knowledge should thus be shared through collaboration and face-to-face communication as opposed to other more explicit mediums.

3.2.7 Principle 7

7) Working software is the primary measure of progress.

The role that software plays in an agile project is mentioned in the review of the first principle, where Fowler proclaims that when measuring the success of projects, the yardstick is business value, i.e. how valuable is the delivered software to the customer. This viewpoint is carried further in the seventh principle. When measuring progress, software is the prime artifact to use. To understand the principle it is necessary to identify what *"working software"* entails and what it means to place it as the primary measure of progress.

Fowler describes the agile view on working software:

> *"These working systems are short on functionality, but should otherwise be faithful to the demands of the final system. They should be fully integrated and as carefully tested as a final delivery."* [34]

The concept *"Working software"* seems to involve two aspects. It must be working (fully integrated and tested) and it must uphold business value (faithful to the demands of the final system). Highsmith [4] mentions, that at every iteration the customer decide whether business value has been delivered or not. If value has not been delivered, the project is either canceled or corrective action is taken. If value has been delivered the project continues [4]. Fowler explains why *working software* means software should be tested and integrated:

> *"The point of this is that there is nothing like a tested, integrated system for bringing a forceful dose of reality into any project. Documents can hide all sorts of flaws. Untested code can hide plenty of flaws."* [34]

The flaws hiding in documents, untested code or code that is not integrated in the full system, means that these artifacts are poor measures of actual progress. Fowler continues and points out that when measuring progress, it is necessary to get all the important factors right. If anything is missing, the "doers" will most certainly alter what they do in order to produce the best result. E.g. if using lines of executable code as a yardstick for progress, it will invite developers to write lots of code in order to produce the best measure. If the value of the code is not paid any attention, this kind of measurement would produce misleading results according to Fowler.

Agilist argue that only working software can tell anything true about the state of a project. Progress is only made when a piece of the final system is delivered to the customer. Highsmith elaborates on the differences between delivering documentation-like artifacts and delivering business value:

> *"If we cant show demonstrable, tangible progress to the customer, then we dont really know if we are making progress. Data models, UML diagrams, requirements documents, use cases, or test plans may be markers that indicate some degree of progress (or at least activity), but these markers are not primary measures. Features that implement a portion of a business process, features that demonstrate that a piece of flight avionics software actually drives the required actuators, features that allow a doctor to extract and view a digitized CT scan – these are primary measures. Nothing else qualifies."* [4].

Highsmith and Cockburn recognizes that other kind of measures are permitted, but *"working code is the one to bank on."* [3].

It is clear that the seventh principle supports the second valued statement as illustrated in Figure 3.8.

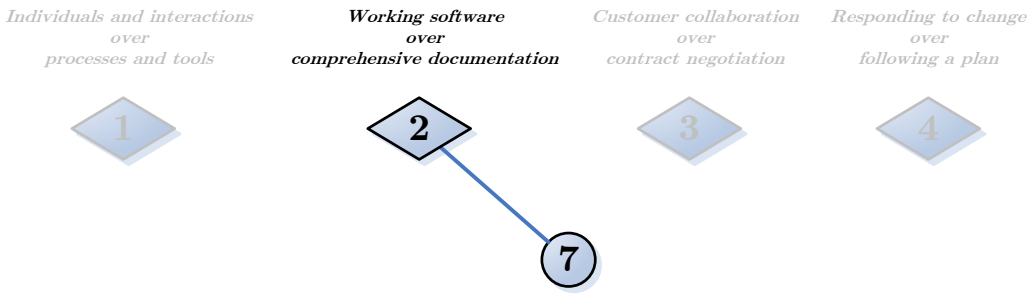


Figure 3.8: The values which principle seven supports.

The review firmly establishes that agile progress control relies heavily on tested and integrated features, which are faithful to the final system. In the agile view, this is the only way the truth about the project state can be expressed, as many other kinds of artifacts such as diagrams, requirements, test plans, etc. can hide flaws.

3.2.8 Principle 8

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

According to Cockburn this principle concerns two dimensions. One relates to social responsibility and the other to productivity. Cockburn [3] explains, that not every manifesto author would sign onto the social responsibility, but all agree on the matter of productivity. Maintaining a sustainable pace throughout the entire project means that individuals on a team should not work more than able to and thereby minimizing overtime. Beck writes:

> *"Work only as many hours as you can be productive and only as many hours as you can sustain. Burning yourself out unproductively today and spoiling the next two days' work isn't good for you or the team."* [2]

The point of this is that developers will not be more effective in their work simply by staying later and working more. Instead they will tire and as a result remove value from the system. Cockburn states that long hours, should they occur, *"... are a symptom that something has gone wrong with the project layout."* [3].

The ninth principle supports the first of the valued statements and is shown in Figure 3.9.

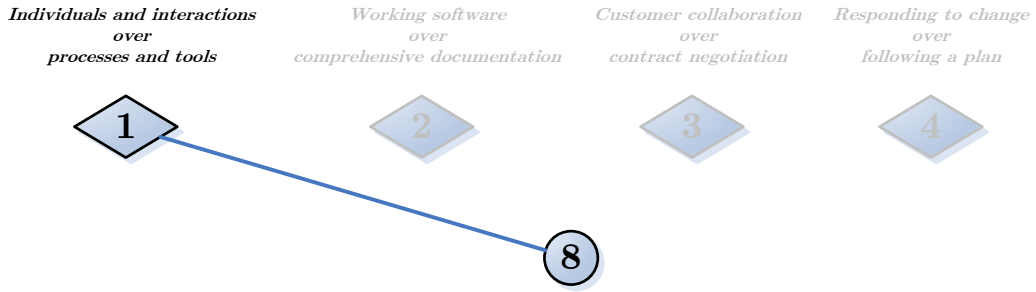


Figure 3.9: The values which principle eight supports.

In the eighth principle, people are not viewed as machines able to stay turned without serious drawback. Though not all authors would sign on to a social responsibility, they all agree to the more romantic perspective on people, their abilities and constraints.

3.2.9 Principle 9

9) Continuous attention to technical excellence and good design enhances agility.

The ninth principle claims that continuous attention to the values *"technical excellence"* and *"good designs"* improve agility. Basically this illustrates the view that good designs, which does not introduces any unnecessary constraints, allow changes to be implemented, thus being agile. The same goes for *"technical excellence"*. Code of high quality is easy to read, modify and find/fix bugs and is therefore more tolerant to changes. However, opponents often criticize Agile Software Development as an excuse for hacking or cowboy coding [16]. To some extent, this viewpoint is proper derived directly from the Manifesto values, were process, planning and documentation is deemphasized on behalf of people, software and adapting to change. Also, as XP advocate a evolutionary design approach where *simple design* and *just enough* are keywords, the viewpoint that Agile Software Development is equivalent to "code and fix" is often assumed. When discussing XP and its design activities, Fowler[17] confirms that evolutionary design in its common usage is a disaster. Allowing the system design to emerge only as a part of the programming process, will evidently produce a poor design and introduce code where bugs are harder to find and kill. Designing up-front attempts to counter this by thinking in advance, plan the design carefully and then freeze it. However, this also introduces headaches, as it is impossible to figure out all the issues that programmers would have to deal with and therefore developers will often have to code around the design. The reason XP still advocates an evolutionary design approach rather than up-front design, is, according to Fowler, that XP includes "enabling practices", i.e. practices which makes the evolutionary design approach work. These includes *Test-first*, *Continuous Integration* and *Refactoring* [17].

Highsmith explains how the these practices makes agile methods able to maintain a high level of technical excellence and good designs:

> *All Agile Software Development Ecosystem's call for frequent testing in order to deliver working software each iteration. XP's frequency is near instantaneous in its test-first mode of operation. Agile approaches call for constant integration testing and frequent builds. Agile approaches call for acceptance testing with customers, both with test cases (XP) and customer focus groups (ASD). Pair programming can be viewed as the equivalent of instantaneous code inspections. Code inspections are a key part of FDD and are advocated by other Agile approaches.* [4]

Fowler, Highsmith and Cockburn agree with the necessity of many of the practices of XP in order to pay attention to good design and maintain a high level of technical excellence. Enhancing agility is equivalent to allowing changes to occur and in the agile view this is the opposite of following a plan and the ninth principle therefore supports the fourth valued statement in the Manifesto.

This is illustrated in Figure 3.10.

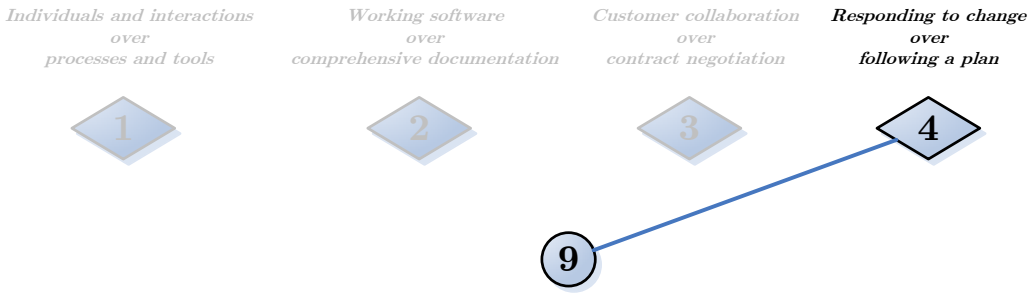


Figure 3.10: The values which principle nine supports.

3.2.10 Principle 10

10) Simplicity--the art of maximizing the amount of work not done--is essential.

The point expressed by this principle is also found in popular phrases such as "YAGNI" (Ya Ain't Gonna Need It) and "KISS!" (Keep It Simple Stupid!). Simple designs and code are easier to understand, debug and maintain. When discussing simplicity in software development, Highsmith refers to three different dimensions: *"Simplicity as minimalism"*, *"Simple design"* and *"Generative rules"* [4]. The first dimension, Highsmith argues, is directed at the tenth principle. The second refers to simple designs as being good designs, as discussed in the review of principle nine. The third, *Generative rules*, apply to the idea of self-organizing teams mentioned in the review of principle five and eleven.

Simplicity as minimalism means to do less of the activities which does not deliver business value and focus on those which does. Higsmith explains how this relates to agility:

> *"Simplicity affects our ability to change, to be adaptable or responsive. Responsiveness relates to inertia. Heavy things have more inertia and are therefore harder to change."*

When discussing simplicity and changeability Highsmith and Cockburn speak of embellishments, i.e. activities which creep into projects because someone think that the activity "should" be useful, but in fact is not. As time goes more of these activities are implemented and never removed and the process grows heavy and thereby looses changeability [4]. As an experiment Highsmith propose that one should try to eliminate activities and documentation until things begin to unravel and then stop reducing. This refers to the teams ability to reflect and tune their process.

The principle thus supports the fourth valued statement and is illustrated in Figure 3.11.

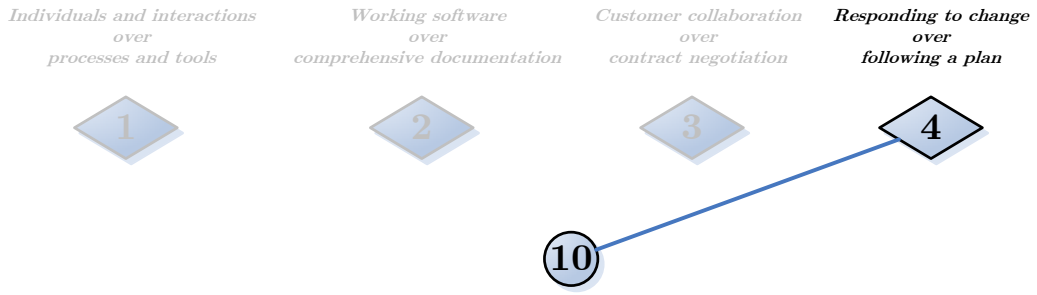


Figure 3.11: The values which principle ten supports.

One of the values in Beck's XP is *Simplicity*. However, Beck does not divide the value into different concepts as Highsmith, but uses it mainly when addressing the system being build, i.e. keeping the design simple. Beck argues, that some designing up-front is necessary, but it should be just enough to be successful and nothing more [2].

3.2.11 Principle 11

11) The best architectures, requirements, and designs emerge from self-organizing teams.

Individuals plays an important role on an agile project, as the first value of the Manifesto indicates and as the review of the fifth principle illustrated. The eleventh principle clearly puts self-organizing teams in the number one position of organizational structures. The argument is that teams which are not locked by numerous rules and policies and which are able to influence their own work process will perform better than more managed teams. However, this does not entirely remove standardization of what people should do as Highsmith states:

> *"... we cannot forget that emergent results live at the edge of chaos, at the balance point between structure and flexibility. The structure, which protects projects from chaos, rests on skills and professionalism. A generative organization provides teams with a simple set of rules and an environment conducive to constant interaction in order to generate innovative results."* [4]

Agile projects thus need to be on the edge, i.e. balancing between order and chaos. The question is how much chaos and how much order is required to be agile? Cockburn addresses this when he states that there is a lot of discussion in the agile community about how self-organizing teams should be.

Beck [2] reform the managers role from being a controller to that of a facilitator. Planning is an activity which the whole team takes part in, while the managers act as *team historians* who communicate progress to the team and external stakeholders and facilitates communication to and within a team. E.g. the manager is not intended to act as a proxy, but introduces suppliers, customers, users etc. to the right person on the team.

Fowler [34] states, that only the developers (the team) can and should decide which process to follow. Attempts to impose a process on a team will lead to rejection. However, management still plays a role as the sharing of responsibility must be equal.

It seems that the common ground the agile authors share on the matter of self-organizing teams, is that teams should be allowed a high degree of freedom and be able to choose and change their process. The eleventh principle is supporting the first of the valued statement as the concept of self-organizing teams stems from a very people centric viewpoint contrary to an imposed process. This is illustrated in Figure 3.12.

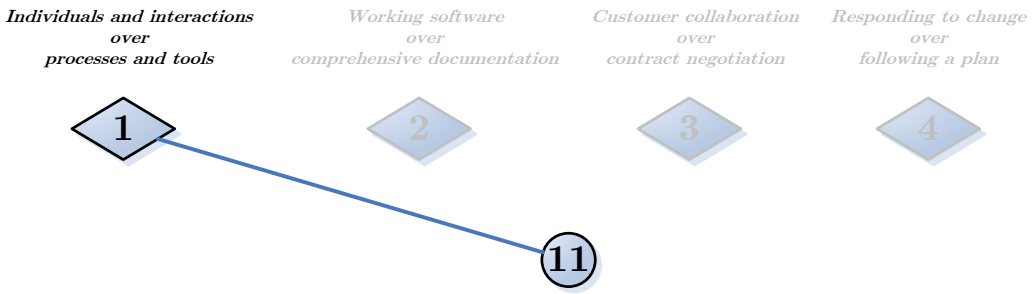


Figure 3.12: The values which principle eleven supports.

3.2.12 Principle 12

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

The last of the 12 principles also touches upon the idea of self-organizing teams. This is explained by Fowler, who notes that self-organizing teams must not only choose their process, but also tune and optimize it as the project proceeds and experience is gained. According to Fowler, the team not only should be allowed adaption, but it is their responsibility to do so. The principle touches on the diversity of software development projects. A method, agile or otherwise, never fits perfectly with a project, its team and organization. It is necessary to change it, in order to make it fit. Beck and Cockburn respectively notes:

> *"Quarters are also a good interval for team reflection, finding gnawing-but-unconscious bottlenecks. You can also propose and evaluate long-running experiments quarterly."* [2]

> *"Bother to reflect on what you are doing. If your team will spend one hour together every other week reflecting on their working habits, you can evolve your methodology to be agile, effective and fitting."* [3]

Highsmith agree on the responsibility and importance of team reflection and adjustments, but also warns about making changes without careful thinking and experimenting:

> *"The 12 XP practices didn't just pop out; they arose from considerable experimentation. Equal experimentation would be necessary to change them. For example, it might seem reasonable – since pair programming has been compared with very short-cycle code reviews – that code reviews could be substituted for pair programming without harm. While this may be true, there is a significant possibility that this substitution would have a negative impact on group collaboration, simplicity of design, and more."*[4]

The important part of reflection is, that it has to be done at regular intervals and not only something, which can be done when there is a need to do so. When widening the fourth and last of the valued statements to include the need to change (adjust) the process, the principle can be seen as supporting it. In addition this "need to change it" stems from a people centered viewpoint and therefore the principle also supports the first of the valued statement. This is illustrated in Figure 3.13

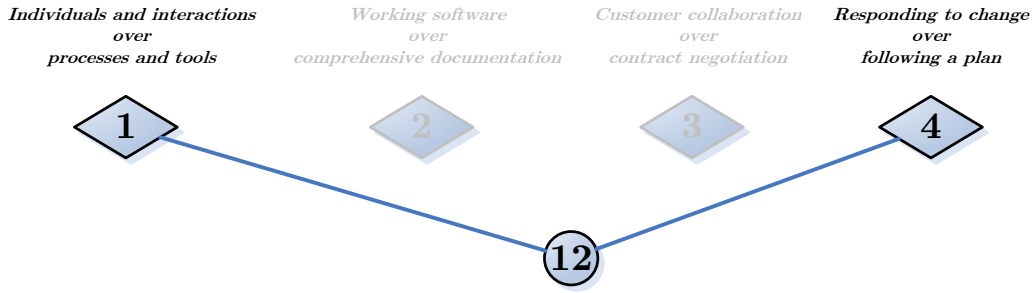


Figure 3.13: The values which principle twelve supports.

3.3 Seven Measurable Areas of Agility

The review of the agile values and principles resulted in the identification of seven measurement areas with 2 important abilities 1) each area is clear enough to measure a distinction between what is agile and what is not, 2) together the seven areas are believed to cover the central points of what Agile Software Development entails.

As with the principles and values, the connection between measurement areas, principles and values is illustrated using the symbols shown in Figure 3.14

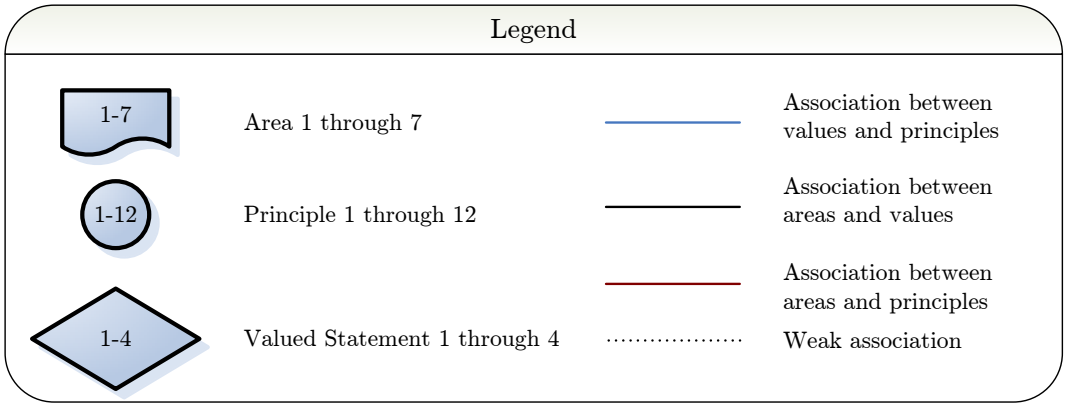


Figure 3.14: Circles are used to describe principles where the number inside the circle refers to the principles number (e.g. principle 1). Diamonds describes a valued statement and the square-piece is the symbol used for the agile concepts. A connection between values and principles indicates that the principle supports the value and a connection between a concept and a value or principle indicates that the concept is drawn from the value or principle respectively.

3.3.1 Organizational Structure

The structure of an organization refers to the way interrelated groups in an organization are constructed. The agile values and principles clearly speak in favor of a non-bureaucratic structure. The question, especially for practitioners, is be how self-organizing should teams be? Even the agile authors does not entirely agree on this matter as mentioned in principle eleven. Agile Software Development does not claim that total chaos is superior to order. It all depends on the situation. What can be said with more certainty is, that the agile view on organizational structure is that teams (i.e. developers, business people, etc.) need a certain level of freedom. The important thing is that a team is able to choose and define their own development process. One of the values in the Manifesto states that *people* is valued over *process* and therefore clearly abandons a mechanistic world view. The concept of a *self-organized* team should thus be seen as opposed to *managed teams*, where policies are constructed by upper management and followed by the team. When measuring agility in this area, respondents attitude towards these to opposing concepts can be assessed.

The correlation between the values and principles and how this measurement area relates to them can be seen in Figure 3.15.

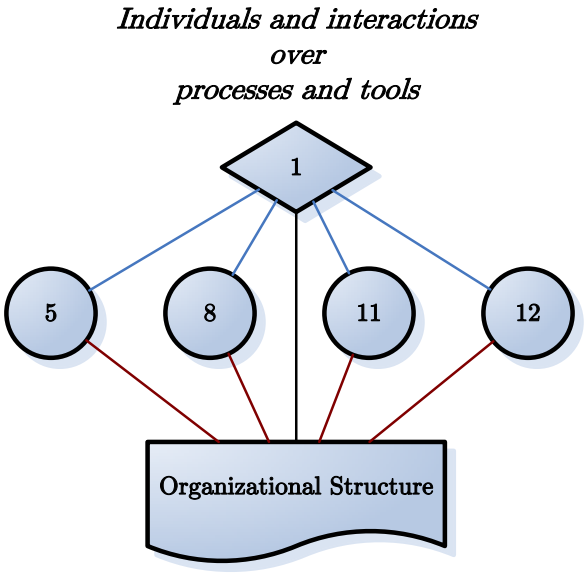


Figure 3.15: The correlation between the values, principles and the *organizational structure* measurement area.

3.3.2 Requirements Handling

Gathering requirements and distributing them to the development team for implementation is two very central activities in all software projects. However, the agile way of carrying out both activities differ significantly from plan-driven development. The review of principles one and four are connected to this area as well as the third valued statement. This relationship is depicted in Figure 3.16

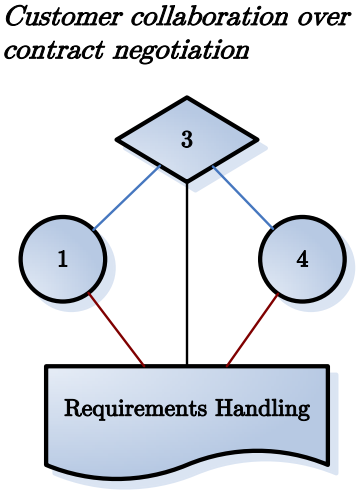


Figure 3.16: The correlation between the values, principles and the *requirements handling* measurement area.

In the review of principle one it is established that delivering customer value is the highest priority to agilists and therefore continuous customer collaboration is essential. The fourth

principle further expresses the view that developers need access to people with business knowledge in order to understand what to build. In addition, business people need to interact with the developers to understand what presently have been build and decide the next move. This view on *continuous customer collaboration* is a unique and central agile concept. It should be seen as opposed to plan-driven requirements gathering and requirements distribution. Here requirements might be gathered by communicating with the customer, however, focus is on the requirements specification and not customer collaboration or interaction with business people. When measuring agility in this area, respondents attitude towards *continuous customer collaboration* and a *requirements specification* can be assessed.

3.3.3 Project Progress Control

Estimating the current status of a project is important not only in agile projects, but also plan-driven. The agile way of measuring progress relies on software as the primary measure. As described in the review of principle seven, this approach should especially be seen as opposed to the use of documentation. The correlation between the values and principles and this measurement area relates to them is seen in Figure 3.17.

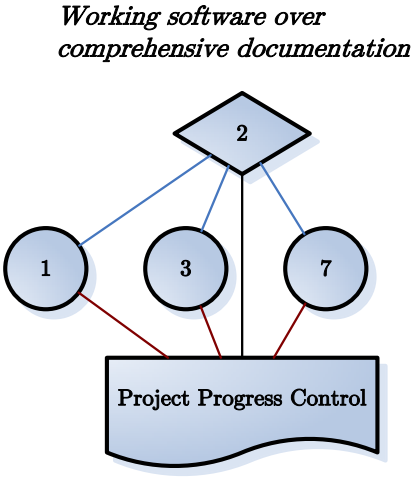


Figure 3.17: The correlation between the values, principles and the *project progress control* measurement area.

In the review of the principles concerning *working software* it is also seen that *working software* refers to integrated and tested features. When measuring agility in this area, respondents attitude towards *working software* and *documentation* can be assessed.

3.3.4 Software Development Model

The most significant concept in Agile Software Development is iterative development. All agile processes must be iterative. It is not possible to say exactly how iterative agilists believe is appropriate, as this depends on the situation. However, it is clear from the review that timeboxing-techniques and small software increments are important to all agile methods.

Timeboxing and keeping increments small allows iterations to be delivered early and frequently. This kind of iterative development is best illustrated when contrasted with a more plan-driven approach to software development, where the process is divided into sequential phases. The correlation between the values and principles and how this measurement area relates to them can be seen in Figure 3.18.

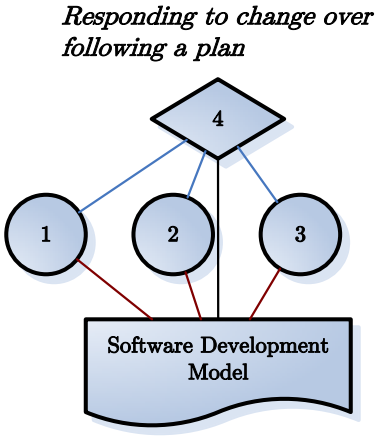


Figure 3.18: The correlation between the values, principles and the *software development model* area.

When measuring agility in this area, respondents attitude towards *iterative development* and a *plan-driven development* can be assessed.

3.3.5 Project Success Factors

A thing which clearly distinguishes Agile Software Development from plan-driven, is the viewpoint on what constitutes success in a software project. For agilists the only thing which matters is delivering business value and thereby satisfying the customer. Principle one is properly the best indication of this. The review of the principle makes it clear that other factors, such as keeping budget and keeping deadline are measures which agilists disapprove of. The correlation between the values and principles and how this measurement area relates to them can be seen in Figure 3.19.

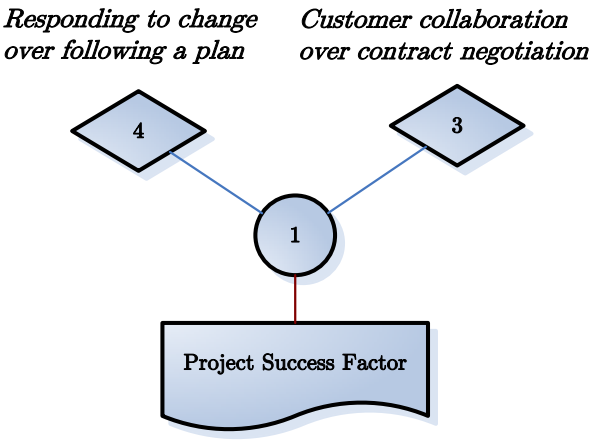


Figure 3.19: The correlation between the values, principles and the *project success factors* area.

When measuring agility in this area, respondents attitude towards *delivering business value* and being *on time and on budget* can be assessed.

3.3.6 Design

The review of the principles shows that the design activity in Agile Software Development can be characterized as focusing on a simple design which solves the present problems and nothing more, as opposed to a comprehensive design approach where the future is accounted for. This is illustrated in Figure 3.20, where the measurement area of design is connected to principles nine and ten, and connected to the fourth value.

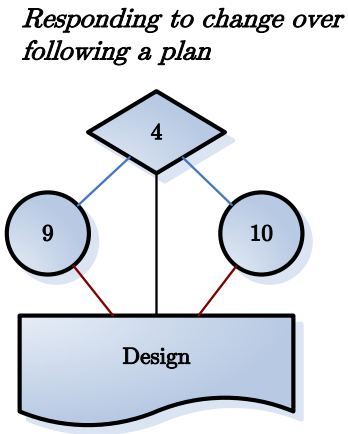


Figure 3.20: The correlation between the values, principles and the *design* area.

When measuring agility in this area, respondents attitude towards *simple design* and a *comprehensive design* can be assessed.

3.3.7 Knowledge Distribution

Distribution of knowledge refers to how knowledge is distributed to and among team members. This, not only concerns gathering and distributing requirements as covered by the *requirements handling* area, but how the team transfers and maintains knowledge. Agilists prefer to use tacit knowledge, i.e. knowledge is not written down but shared among all the team members through collaboration and face-to-face communication. The correlation between the values and principles and how this measurement area relates to them can be seen in Figure 3.21.

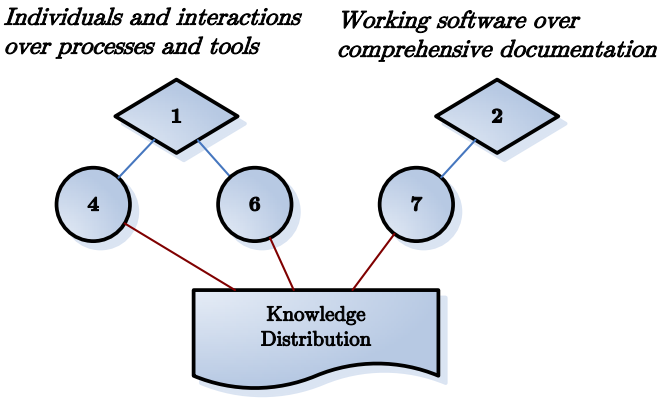


Figure 3.21: The correlation between the values, principles and the *knowledge distribution* area.

This does not imply that knowledge can not be written down, but that as a rule tacit knowledge is preferred over explicit. The plan-driven approach on the other hand advocates the use of documentation to transfer knowledge. When measuring agility in this area, respondents attitude towards *tacit knowledge distribution* and *explicit knowledge distribution* can be assessed.

4 Questionnaire Design and Administration

To measure the agility of IT professionals through a questionnaire basic knowledge is required on questionnaire design and the pitfalls there might exist.

In this chapter we introduce the questionnaire design process, the theory behind validity and reliability of measurement instruments and then we construct the instrument that is used to measure agility in our questionnaire is constructed.

4.1 Design Process

The questionnaire design process is a very iterative process. A number of different techniques and disciplines are applied to develop, tune and validate the final questionnaire sent to respondents. This section provides an overview of the different activities during the design of the questionnaire.

An iterative process involves several benefits. First of all, gathering quantitative data using a questionnaire is not a research method the research group has much experience with. Allowing the questionnaire to evolve over several iterations would result in the identification of several big and small mistake that can be corrected prior to conducting the main survey.

4.1.1 Testing the questionnaires

To develop a good and understandable questionnaire, tests are conducted using external test respondents. Using external test respondents to evaluate work products is very useful.

Respondent/ Characteristics	A	B	C	D
Age	32	27	52	27
Occupation	Student	Student	Associate Professor	Employee in the private sector
Area	Computer Science	Humanistic Science	Head of Pervasive Healthcare Lab	Software Development

Table 4.1: Characteristics for the four different test persons used in the test.

People who are not a part of the research team and not involved in the daily discussions, are able to provide a different viewpoint on issues and problems, which often leads to new ideas and solutions. This is also known used in *Usability Testing*, a concept widely known within computer science and Human-Computer Interaction (HCI). The purpose is to test the usability of e.g. an application by using actual users. This exercise will often uncover any difficulties that a real life user would have with e.g. understanding instructions or interpreting feedback.

The test is also useful when constructing a questionnaire. Words or phrase may be interpreted differently or simply not understood by people outside the research group. Identifying such things before actually deploying the questionnaire is crucial, because it can jeopardize the reliability of the instrument and thus the validity of the research.

Therefore several test respondents are used to review, test and examine the different versions of our questionnaire.

The External Test Respondents

Four different kinds of external test respondents are chosen to participate in the field test. These specific test respondents are selected because of two main reasons: 1) They are easy to access and use, 2) They represent different types of people, with different background and experience levels within software development.

The second reason is important with regard to the value of the tests. It is believed that the survey will reach respondents with very different backgrounds, experience and knowledge, therefore it is crucial that questions are understood correctly, and are interpreted as intended.

The four test respondents and their background is shown in Table 4.1

As the table shows the questionnaire is tested using two students from two different areas of study, an associate professor and a software developer employed in the private sector. Test respondent B studying humanistic science is not familiar with software development, but is used to "stress-test" the questionnaire. The student was not able to answer all the questions

and some were naturally enough hard to understand. However, by conducted the test using a person who is unfamiliar with software development means that the questionnaire is tested under the worst possible conditions.

Test respondent C, who is an associate professor has worked with software development before. This means, that this test respondent can relate to the topic and thus the questions. The respondent can help clarify if the questions asked are understandable and interpreted as intended, by people working with software development daily. Respondent C represents the older part of the target group with many years of experience in software development.

Respondent D is employed in the private sector within the software industry and respondent A, is used as they reflect the younger par of the target population.

Conducting the Field Test

As explained the test is conducted using an approach quite similar to that of usability testing and using the *think-out-loud* protocol that is described by Rubin [5]. The test respondent is given a brief introduction to the research project, the purpose of the test and how the test should proceed. After this, a link to the survey is provided and the test begins.

The test respondent is first asked to read each question out loud and explain how the specific question is understood. If a question is a multiple choice question, the test respondent is also asked to read the different answers out loud and explain these in the same way. Upon having read the question and possible answers, the test respondent is asked to answer the question and explain the answer given.

The test respondents are always asked to state their opinion on the subject first. This can lead to a discussion of questions and concepts that proves difficult to understand. Finally the test respondent is asked to give suggestions to improve the question and the answer choices.

Field Test Findings

The tests were very beneficial. There were many issues with the questionnaire that clearly annoyed and confused the test respondents. This was especially during the first test, where the ordering of questions, inconsistent layout and small textboxes annoyed the respondent. Moreover a significant number of the concepts were clearly interpreted differently by the test respondents. As an example the concept *Process Control and Monitoring* was used in the questionnaire as a non-agile concept. It was believed that the idea of having *process control* and to *monitor* a process was valued in the plan-driven paradigm and de-emphasized in the agile paradigm. However, when asked more specifically about why a respondent placed the concept as a *very important factor to software success*, it was discovered that the test respondent viewed this from different perspectives. The respondent actually argued for a more agile way to control and monitor a process, e.g. by using short iterations with quick feedback, regular reflection meetings, customer involvement, etc. From the viewpoint of the respondent, process control and monitoring could be done in an agile and non-agile way.

The different issue that were pin-pointed by the test respondent during the tests were identified this way and eventually lead to a more reliable questionnaire. Also it greatly improved our confidence in the questionnaire.

4.1.2 Experts and Researchers

In addition to the test respondents, two experts were consulted for advise on the survey and design.

Ivan Aaen is an associate professor at Aalborg University (AAU). He is professionally interested in Agile Software Development and has been for several years and teaches software development methodology. He has also previously conducted a questionnaire survey concerning the use of CASE tools in software organizations. He assisted with different suggestions to the agility measurement instrument and with suggestions on which types of analysis procedures could be applied to our data. Aaen also pointed out some of the possibilities with factor analysis.

Clive Sanford is also an associate professor at AAU who is familiar with using questionnaires as a research tool. He was consulted to provide basic feedback on the design and the questions. He helped improve the questionnaire's understandability.

4.1.3 Gathering Respondents

Collecting useful data through a questionnaire also relies on gathering the right number of respondents and the right kind. There are different ways to select the right respondents and the right number of respondents. Kumar [6] calls this process sampling and he states the following about sampling:

> *"Sampling .. is the process of selecting a few(a sample) from a bigger group (the sampling population) to become the basis for estimating or predicting the prevalence of an unknown piece of information, situation or outcome regarding the bigger group. A sample is a subgroup of the population you are interested in."*

This sampling process has advantages and disadvantages. The advantages are that it is possible to save time, money and resources, because less respondents has to be contacted and administered [9]. But this is gained by compromising the level of accuracy of the findings, because it relies on estimation or predictions about the population. This means that there is a possibility of errors in the estimations [6]. There are two key factors that affects the assumptions drawn from a sample [6]:

The size of the sample: Findings based upon larger samples have more certainty, than those based on smaller ones. As a rule, the larger the sample size, the more accurate the findings.

The extent of variation in the sampling population: If a population is homogeneous with respect to the characteristics under study, a small sample can provide a reasonably good estimate. However, if the population is heterogeneous, a lager sample is needed to obtain the same level of accuracy. As a rule, the higher the variation with respect to the characteristics under study in the study population, the greater the uncertainty will be for a given sample size.

The fist rule is straight forward and simple to understand with regard to certainty in relation to the sample size. However, for the second rule, if the target population is heterogeneous and contains a large variance the sample size has to be increased in order to compensate for the increase in uncertainty.

The importance of selecting the right respondents, sparkled a discussion on how to address this. The target population is clear, because the research questions specifically targets IT professionals. By IT professionals, we mean people who are involved with software development in their organization, e.g. programmers, project managers, etc.

After identifying the target population, the right approach to selecting the respondents has to be made. One of the approaches is to select a specific company and use their IT employees as respondents. However, this will limit the generalizability of the results as the results from one company cannot be directly applied to other companies. So instead of targeting companies and use their IT employees, it is more appropriate to contact an organization with access to a large number of IT professionals. E.g. PROSA or Dansk IT, as they have a large list of members. By using this kind of organizations a great number of respondents is reached. This will also remove the possibility of sample bias, as the sample is not directly specified by us. However, to cover a broad selection of IT professionals more than one organization has to participate.

One disadvantage with using organizations is the lose of control. It is difficult to know who and how many respondents receives the survey, e.g. we do not know who are listed on the members lists and we cannot prohibit respondent from sending the survey to others that friends and collegues. These issues make it difficult to calculate the response rate precisely. The response rate is an indicator of how reliable the results are.

4.1.4 Questionnaire Administration

Kumar [6] mentions different ways to administer questionnaires; *The mailed questionnaire*, *collective administration*, and *administration in a public place*. Each way has its advantages and disadvantages.

The mailed questionnaire: The most common approach to gather information is by mailing the questionnaires to the respondents. The format could be a hard copy or electronically (e.g. e-mail). The main problem with this approach is a low response rate.

Collective administration: One of the best ways of administering questionnaires is to obtain a captive audience, such as students in a class-room or people at a conference.

This approach gives a high response rate and is inexpensive.

Administration in public place: A questionnaire administered at public places, like malls, hospitals, schools, etc. is time-consuming, but it has the same advantages as collective administered questionnaires. The approach and use of either depends on the study and the respondents.

Using the collective administration approach is a good way to get a high response rate at a low cost, however, we do not have access to conferences or class-rooms with IT professionals, so this approach is not an viable option. Also the last approach using public places would not be an option for us, because we specifically target IT professionals. The mailed questionnaire approach is a good way to reach as many respondents as possible, because it is easy to distribute and administer electronically, even though there is a problem with low response rates. However, there is some advantages that will be mentioned later that promotes this approach as being the best way to administer our questionnaire.

Medium

There are two different electronic mediums that can be used; *e-mail* or *online*. Based on its advantages and the target population the online medium is selected. Some of its advantages are that it is extremely fast, no cost is involved once the set up has been completed, it is possible to show pictures, video and play sound, use complex logic, and other features which are not possible with paper questionnaires or most e-mail surveys. Also it can promote understanding by using colors, fonts and other formatting options which are also not possible in most e-mail surveys. More importantly there is no need for manual distribution and recollection of the questionnaires. Besides that the online questionnaire data can be collected electronically using survey software and can be analyzed using advanced statistical software.

However, there are still some disadvantages to using online questionnaires, such as current use of the Internet is far from universal, respondents are likely not to complete a long questionnaire, no control of who attends unless access control is applied for certain respondents, no control over multiple responses, which can all introduce biases.

4.2 Validity

The term *Validity* refers to the conceptual and scientific soundness of a research study [9]. Validity is important because, as researchers we should produce valid conclusions on our findings, and this is only achieved by eliminating or minimizing the effects of extraneous influences that might detract from the actual findings [9]. Kumar states, that to be able to generate some "output", i.e. an conclusion, there has to be some "input". The input comes from the different steps that we have to take, e.i. selecting a sample, collecting data, processing data, the application of statistical procedures and writing a report [6]. The way this is done can effect the quality of our conclusion.

Kumar [6] states, that there are generaly two perspectives on validity:

- Is the research investigation providing answers to the research questions for which it was undertaken?
- If so, is it providing these answers using an appropriate method and procedure?

Validity is the ability of an instrument to measure what it is designed to measure in terms of measurement procedures [6]. Kumar cites Babbie (1990) which states *"validity refers to the extent to which an empirical measure adequately reflects the real meaning of the concept under consideration"*. This means that we have to be able to rule out alternative explanations of the results. There are two ways to establish validity of our research instrument; logic and statistical evidence. The logic procedure concerns establishing a logical link between the question and the objectives, whereas the statistical procedure concerns providing hard evidence by calculating the coefficient of correlations between the questions and the outcome variable [6]. The logical procedure is easy, as it is simple to see a link between the questions and the objective, however, it lacks the statistical evidence to back the justifications made. For instance it is possible to establish a logical link between age, sex, etc. and a question because they are concrete. Whereas a question measuring attitude is not as concrete and it is necessary to ask several questions in order to cover different aspects of the question [6].

4.2.1 Types of Validity

There are three types of validity [6]:

- Face and content validity
- Concurrent and predictive validity
- Construct validity

Face and Content Validity

This concerns the assessment if an instrument is measuring what it is intended to measure, based on an logical link between the questions and the objectives of the study. This type of validity is easy to apply, i.e. each question or scale must have a logical link with the objective. This establishment is called a *face validity*. Kumar [6] states that it is equally important that the items and questions cover the full range of the issue or attitude being measured. Assessment of the items of an instrument is called *content validity*.

Even though it is easy to establish a logical link and thus a logical argument for validity, there are certain problems that we have to take in account [6].

1. The assessment is based upon subjective logic. Hence, no definite conclusions can be drawn and different people may have different opinions about the face and content validity of an instrument.

2. The extent to which questions reflect the objectives of a research study may differ. If the researcher substitutes one question for another, the magnitude of the link may be altered. Hence, the validity or its extent may vary with the questions selected for an instrument.

Concurrent and Predictive Validity

Using a scale in a measurement instrument, the validity of it can be investigated by seeing how good an indicator it is [6]. To validate the scale it can be compared to other assessments. There are two types of comparisons[6]; *predictive* and *concurrent*. Predictive validity is judged by the degree to which an instrument can forecast an outcome. Concurrent validity is judged by how well an instrument compares with a second assessment concurrently done. Kumar further states that the predictive validity usually can be expressed in the terms of the correlation coefficient between the predicted status and the criterion. Such a coefficient is called a validity coefficient.

Construct Validity

Construct validity is based upon statistical procedures and more sophisticated techniques for establishing the validity of an instrument. Kumar states that it is determined by ascertaining the contribution of each construct to the total variance observed in a phenomenon. In our research project we would like to find out the agility of IT professionals. We have to find several factors that can be used to indicate agility and construct question to ascertain the degree to which the respondents consider each of the factors important for agility. Kumar states that the contribution of theses factors to the total variance is an indication of the degree of the validity of the instrument. The greater the variance attributable to the constructs, the higher the validity of the instrument.

4.3 Reliability

The concept of reliability can be seen from two perspectives [6]:

1. How reliable is an instrument?
2. How unreliable is it?

The first question concerns the ability of the instrument to produce consistent measurements. This means that if we use an instrument to measure agility more than once under similar conditions and get the same or similar results, the instrument is considered to be reliable [6]. Kumar further states "the greater the degree of consistency and stability in an instrument, the greater its reliability is".

The second question concerns the inconsistency of the instrument. For instance if we use an instrument to measure agility more than once under similar condition and we get results

that differs. The degree of the differences in the measurements are an indicator of the extent of its inaccuracy. This is a reflection of the instrument's unreliability. The less the difference between two sets of results, the higher the reliability of the instrument [6].

4.3.1 Factors Affecting Reliability

Kumar states that it is impossible to have a research tool which is 100% accurate, because it is impossible to control external factors that affect the reliability. Some of the factors that affect reliability are:

The wording of questions: Unclearness in the wording of a question can effect how the respondents interpret the question, thus affect the reliability of the instrument.

The physical setting: If any change is done to the physical setting in the case of an interview, the respondents may give different responses and thus affect the reliability.

The respondent's mood: The mood of the respondents can affect the reliability of an instrument.

The nature of the interaction: The form of interaction between us as researchers and the respondents can affect the reliability of the instrument.

The regression effect of an instrument: When a research instrument is used to measure attitudes towards an issue, some respondents tend to change their opinion if they feel they have expressed themselves positively or negatively towards an issue, thus affecting the reliability.

There are different methods to determine the reliability of an instrument. The various procedure are classified into two groups [6]:

1. External consistency procedures
2. Internal consistency procedures

External Consistency Procedures

External consistency procedures concern the comparison of findings from two independent processes of data collection. This is used as a means of verifying the reliability of the measure. There are two methods that do this [6]:

Test/re-test: This procedure is a commonly used method to establish reliability of an instrument. It concerns the repeated administration of the measurement instrument under the same or similar conditions. The ratio between test and re-test scores is an indication of the reliability of the instrument. The greater the ratio, the higher the reliability of the instrument. The advantage in using this method is that it permits

the instrument to be compared to itself. The disadvantage is that the respondents may recall the answers they gave the first time, thus affecting the reliability of the instrument.

Parallel forms of the same test: Two instruments are constructed that measures the same phenomenon within two similar populations. Then the results are compared with each other. If they are similar the instrument i reliable. However, there are also some advantages and disadvantages in using this procedure. The advantage is that it eliminates the problem of recall that is present in the test/re-test procedure. The disadvantage is that we have to develop two instruments instead of one, and it is difficult to create two instruments that measures the exact same phenomenon and it is difficult to achieve comparability in two populations.

Internal Consistency Procedures

Internal consistency procedures concern items measuring the same phenomenon should produce similar results. The *split-half technique* is a commonly used method to measure reliability in an instrument. The idea behind it is that it is designed to correlate half of the items with the other half and is appropriate for instruments that measure attitude towards issues or phenomenons. The aim is to divide questions or statements that measure the same issue or phenomenon in two halves. The scores obtained by administrating the two halves are correlated. Kumar states that the reliability is calculated by using the product moment correlation between the scores obtained from the two halves. The product moment correlation calculates the reliability on the basis of only half the instrument, thus it need to be corrected to asses the whole instrument. This is called the stepped-up reliability and the Spearman-Brown formula is used to calculate the reliability of the whole instrument [6].

4.4 Instrument for Measuring Agility

Measuring agility in a questionnaire depends on the instrument used for the measurement. Our questionnaire seeks to measure agility through the use of respondents' attitude. Chapter 3 took the term agility and operationalized this through the Manifesto's values and principles. The result of this operationalization is seven measurement areas that cover, what agility entails and makes it possible to distinguish between agile and non-agile respondents. These measurement areas are further developed into a set of questions that comprise the instrument to measure agility.

4.4.1 Attitudinal Scales

To measure the attitude of respondents, attitudinal scales are used. These scales originate from psychology where they are used to measure abstract concepts that are otherwise "unmeasurable", e.g. authoritarianism and self esteem [41].

Two of the most common scales are the *Likert Scale* and the *Thurstone Scale*.

The Likert Scale

The *Likert scale* is also known as summated rating. In contrary to e.g. the *Thurstone scale* it assumes that each statement/item on the scale is of equal weight or importance. The respondent is asked to rate each statement on a response scale. Response scales can vary in length and can be of even or uneven length, e.g 1-5, 1-9, 0-5, etc. Using a response scale of uneven length (e.g. 1-5), the respondent is allowed to choose a middle or neutral value (3). However with an even response scale (0-5) the respondent is forced to make a choice.

Using a the *Likert scale*, the final score for each respondent is calculated by summing up the ratings for each statement on the scale. The final score is not useful by itself, but helps relate respondents and their scores to each other. For example, determining if respondent *A* is more happy with his job than respondent *B*.

The Thurstone Scale

One way of calculating an overall score for e.g. respondents' attitude towards their new job, is to use the *Thurstone scale*. The scale is named after one of the first and most productive scaling theorists, Louis Leon Thurstone [41]. The scale is developed by using a panel of judges to determine the value or weight of each statement or item on the scale. With these metrics it is possible to calculate an overall score of peoples attitude towards a concept or phenomenon.

4.4.2 Question Types

To decide which types of question to use, a brainstorm was done, where different kinds of question types were proposed. Initially four types of questions were identified in order to asses peoples agility.

1. Criticality questions
2. Agree or disagree questions
3. Valued statements
4. Bipolar preferred questions

All four types of questions are believed to be useful for measuring peoples agility but with different properties. The four approaches are discussed, but in order to assess them further prototypes of all four is made and their applicability is tested. These tests are conducted as *think-out-loud* sessions, where each test respondent, while answering the questionnaire, explained what she understood by a question and what was unclear. The *think-out-loud* session is explained in Section 4.1

Criticality Questions

The criticality questions focus on the different concepts that the agile and traditional paradigm relate to project success. E.g agile concepts include *customer involvement*, *iterative development*, etc. whereas traditional concepts include *documentation*, *managed teams*, *phase testing*, etc. The respondents will have to state how important each concept is in order to achieve success in a software project on a *Likert scale*. An agile respondent will then rate the agile concepts as important and the traditional concepts as non-important. This will give each respondent an agility score and thereby make it possible to measure the respondents' agility.

Agree or Disagree Statements

Using the *Thurstone scale* the objective is to develop a set of statements concerning peoples attitude towards software development values and principles. Each statement will phrase an agile or traditional value that the respondent will agree or disagree with. E.g. "*Frequent customer contact is essential?*". The set of statements is then valued or weighted according to how agile the statements are, by a group of individuals. In this way a score assessing each respondents' agility is possible to gather.

Valued Statements

The valued statements approach originate from the fact that most values only become agile when they are weighted over another value. This is also what the Manifesto uses to communicate the key values in Agile Software Development. When used to measure the respondents' agility each agile value is weighted over a non-agile value and the respondent states if she agrees with this valued statement.

Bipolar Preferred Questions

The bipolar preferred question type takes advantage of the same duality as with the valued statements. However, instead of phrasing it at a statement that one can agree or disagree with, it is phrased as a question. Two opposing concepts are placed on either end of a scale, where one concept is agile and the other is not. The respondent is then asked which of the two concepts she prefers and rates it on the scale. Additionally a situation or context can be introduced in the question formulation. This would give the respondent additional background information which makes it possible to understand the question and thus making it possible to answer. This approach makes this type of questions very effective and flexible for our purposes.

Choice of Measurement Instrument

The criticality questions is fairly easy to construct using a Likert scale, however, the value of the answers is questionable. Some of the test persons misunderstood questions that lead to answers that did not reflect their true attitude towards Agile Software Development. The test persons is also confused by the fact that different concepts are important in different situations and contexts.

The *Agree or Disagree* statements yielded valuable results, but was difficult to construct without introducing biases. For the statement to reflect an agile or non-agile value it must be phrased either positively or negatively. For example, the statement *"Short development iterations are essential"* sets the agile value of short development iterations in a positive light. This could potentially introduce a bias to one side.

The valued statements have the advantage of being paired, which eliminates the need for a positive or negative phrasing, and also more effectively communicates the underlying values of each question. However, compared to the bipolar preferred questions they still lack the flexibility of providing a context they can be valued in.

The bipolar preferred questions proved to be the most effective. The responses from the test respondents were favorable and the questions were interpreted correctly. Also the strength of this question type is that the respondent is forced to make a choice between an agile and a non-agile concept.

Of the four different question types, the most effective way of measuring the respondents' attitude towards Agile Software Development is through the use of bipolar preferred questions.

4.4.3 Four Different Perspectives

The research questions require that not only the agility of respondents is measured but also the agility of three other groups or perspectives. The other three perspectives are team, organization and customer.

Given that the information cannot be collected directly from a respondent's team, organization and customer, the measurement instrument must be able to assess the agility of these three additional perspectives indirectly. This is done by asking all the respondents to judge what they mean their team, organization and customer prefer on each question. This approach has the drawback of relying on the respondents' subjective judgement and may therefore not reflect the true nature of the team, the organization and the customer, with regard to their attitude towards the values presented in the questionnaire. However, it will provide a rough picture of the tendencies for the three additional perspectives.

4.4.4 From Measurement Areas to Questions

Even though Agile Software Development has been operationalized to yield a respondent's agility, through the measurement areas introduced in Chapter 3, there are still several things that must be taken into consideration. Specially with regard to validity and reliability mentioned in Section 4.2 and Section 4.3.

1. Is the question understandable?

2. Does the question reflect the most important aspect within the measurement area?

3. Does the question differentiate between agile and non-agile concepts?

All these criteria must be considered for each question in order to maximize the usefulness of the questionnaire and the instrument used to measure respondents' agility. This also has to be done with validity and reliability in mind. The first criteria is evaluated on the basis of the test persons' answers and their evaluation of the questions. This evaluation results in a minor adjustments to enhance understandability. The two final criteria reflects the operationalization of Agile Software Development done in Chapter 3 and is also evaluated based on the test sessions in Section 4.1.

We identified seven measurement areas in Section 3.3. These areas will be discussed and transformed into question in the following subsection. The entire questionnaire can be found in Appendix B.

Organizational Structure

The measurement area *organizational structure* springs from the agile principle of self-organizing teams and the values of people centric software development. This is set up as a bipolar preferred question, with *self-organizing* teams at one end and *managed teams* at the other. Figure 4.1 shows the question as it is presented in the questionnaire. The top question introduces the area and asks the respondents to rate which concept they prefer on the bipolar scale. This question is then asked for all four perspectives.

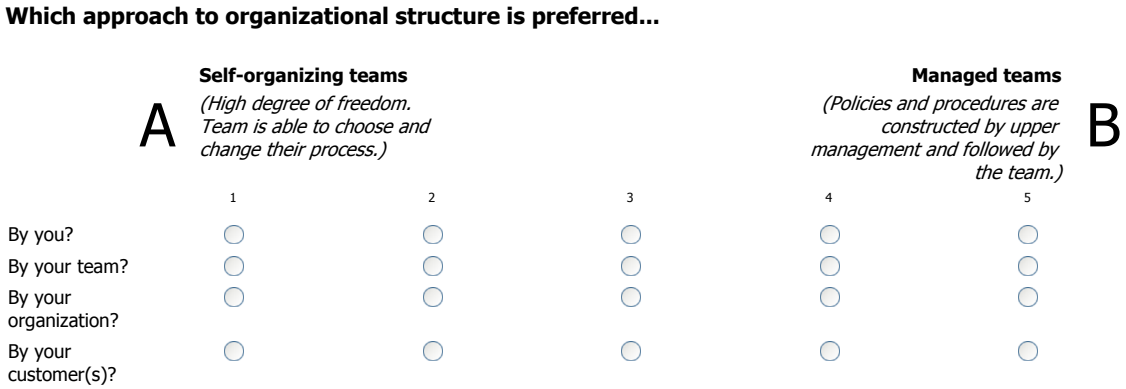


Figure 4.1: The question inferred from the measurement area Organizational Structure

Figure 4.1 also includes a brief description of the concepts at each end of the scale. This is done to ensure that the respondents understand the questions in the same way, avoid misinterpretations and clarify the concept. Thus increasing the reliability and validity of the questions.

The respondents have to mark their choice for each of the four perspectives on the scale from 1 to 5. If the respondent chooses 1 is the respondent prefers the agile concept of self-organizing teams whereas if 5 is chosen the respondent prefers the non-agile concept of managed teams. This is how the scale differentiates between agile and non-agile respondents. The characters A and B are used as reference points for a generalized example that should help the respondent understand and interpret the question in the same way through all the bipolar preferred questions.

Requirements Handling

Requirements handling is the measurement area which centers around the agile value of relying on *continuous customer collaboration* throughout the development project. This concept is flanked by the more plan-driven approach of handling requirements through a *software requirement specification*. These two concepts describe the agile and non-agile way of handling requirements as explained in Chapter 3. This is illustrated in Figure 4.2 where each concept takes its place at either end of the bipolar preferred scale.

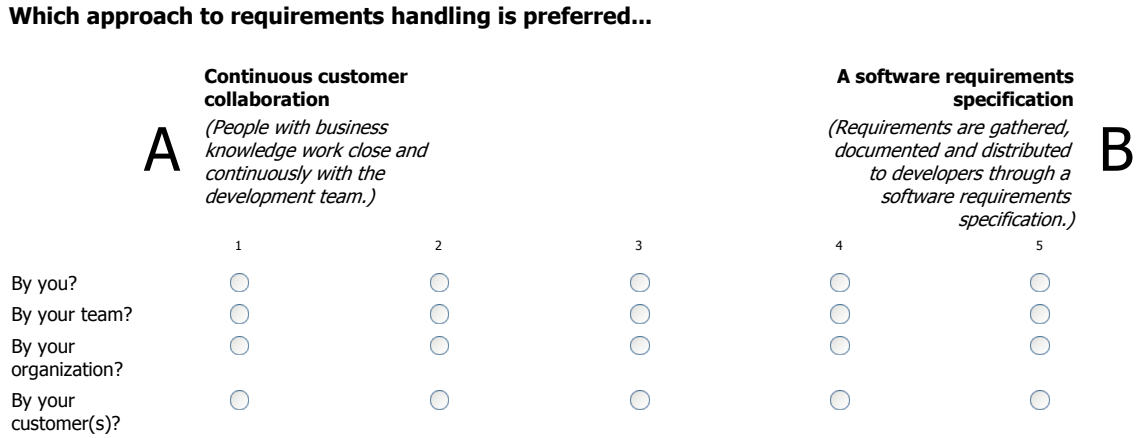


Figure 4.2: The question inferred from the measurement area Requirements Handling

The descriptive text beneath the agile concepts further includes the agile principle of relying on people with business knowledge to make all business decisions, and relay this information through close and continuous collaboration with the development team. This is the essence of this concept and stands in good contrast to the regimented upfront requirement specification phase of plan-driven approaches.

Project Progress Control

The *project progress control* measurement area includes the agile value of *working software* over *comprehensive documentation*. The question sets of in the agile principle that the only measurement of progress is working software. In contrast to this agile principle, the plan-driven approach relies on documentation to control and measure project progress. The question in Figure 4.3 clearly shows the two opposing approaches.

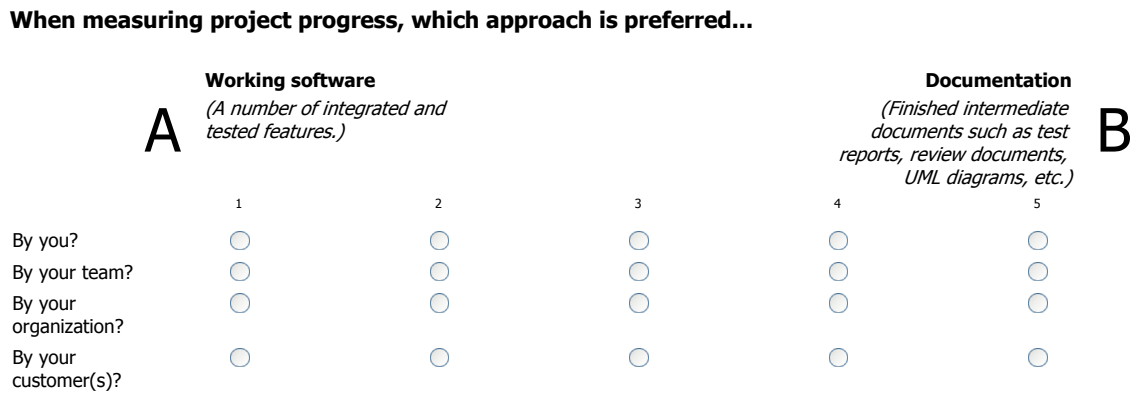


Figure 4.3: The question inferred from the measurement area Project Progress Control

To avoid misunderstandings a brief description states what should be interpreted as working

software and documentation. To avoid negative phrasing we did not use the term *comprehensive* before documentation. But in the text beneath it is described what "documentation" is covering.

Software Development Model

The *software development model* measurement area lies at the heart of Agile Software Development. The area springs from the values and principles that all point towards an *iterative development* approach as the best way to handle changing requirements. The iterative approach can be seen in contrast to the *plan-driven development* approach that relies on the waterfall model. Figure 4.4 shows the question from the questionnaire with the bipolar preferred question and the four perspectives.

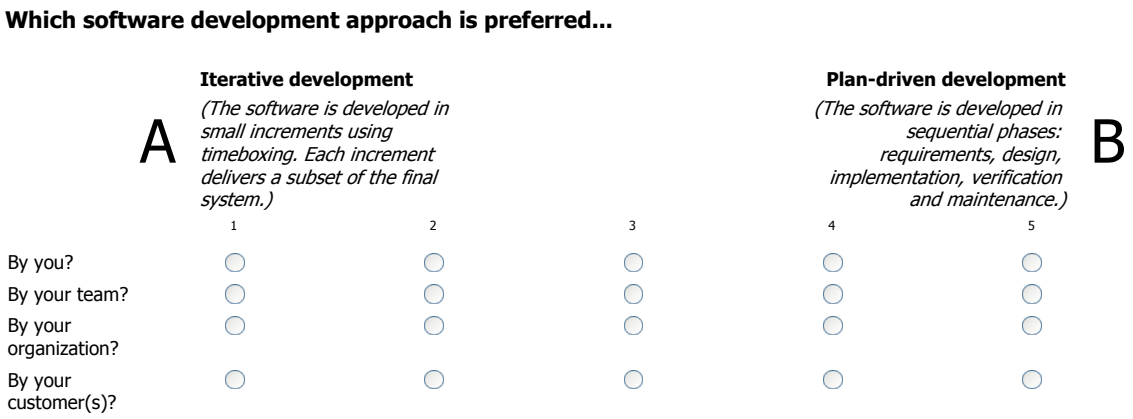


Figure 4.4: The question inferred from the measurement area Software Development Model

The agile concept *iterative development* is derived from the description of the measurement area and covers the important techniques of timeboxing, small increments and that each increment is a subset of the entire system.

Project Success

The *project success measurement* area relies on the principle of satisfying the customer. In the principle this is achieved through continuous delivery of working software, but after a more detailed study of the principles the concept of *business value* emerged. To satisfy the customer, the customer has to get something that is of more value than what they put in. This view does not reflect the plan-driven approach of measuring success, which is based on *deadlines and budget* constraints. These two opposing approaches are shown in Figure 4.5.

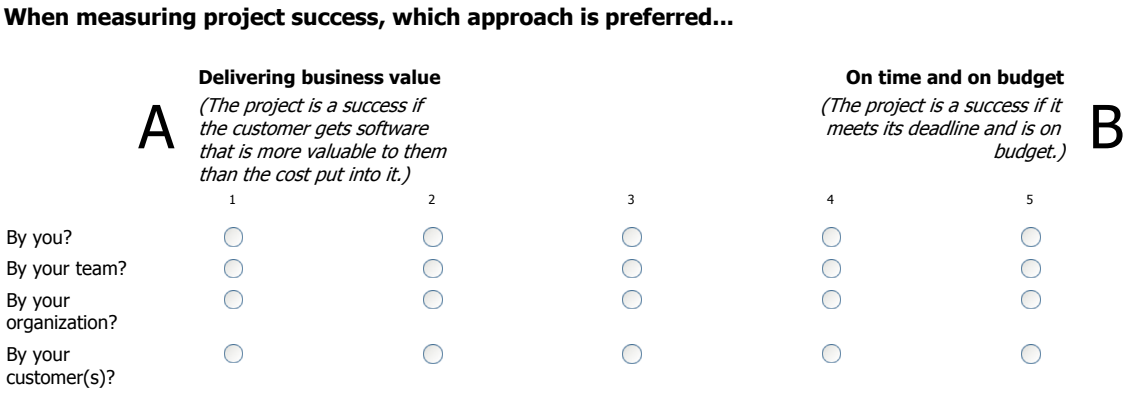


Figure 4.5: The question inferred from the measurement area Project Success

Design

The *design* measurement area is derived from the agile principle of striving to keep things as simple at possible. This simplicity promotes transparency, code refactoring and maintenance. This question focuses on *design simplicity* as opposed to a *comprehensive design* that takes possible future features into account. These two sides reflect the agile approach to software design and the plan-driven approach. Figure 4.6 shows the preferred question as it is presented in the questionnaire.

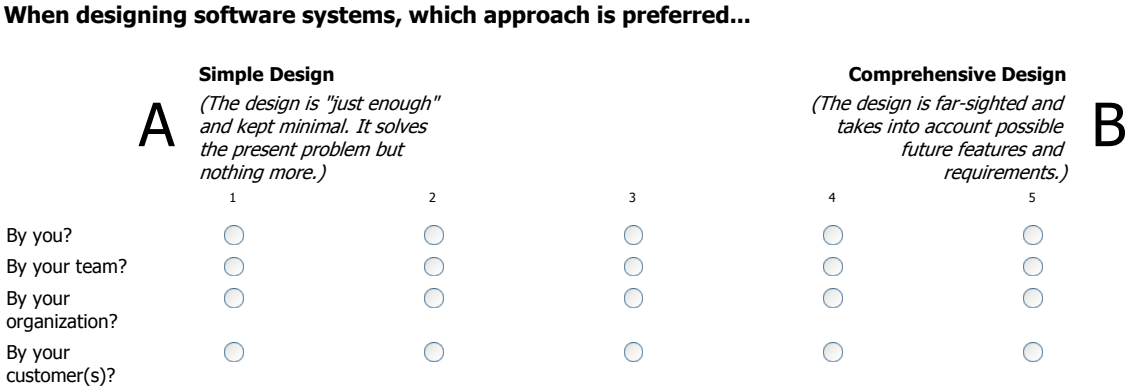


Figure 4.6: The question inferred from the measurement area Simplicity

Knowledge Distribution

This measurement area focuses on the agile principle of *Tacit face-to-face communication*. Through the process of operationalizing the agile values and principles this communication was discovered to be essential when distributing knowledge within the project team as a whole. On the other end of the bipolar scale the plan-driven approach to knowledge

distribution is done via *explicit documentation*. These two opposing concepts cover the measurement area of knowledge distribution and distinguishes between agile and non-agile respondents.

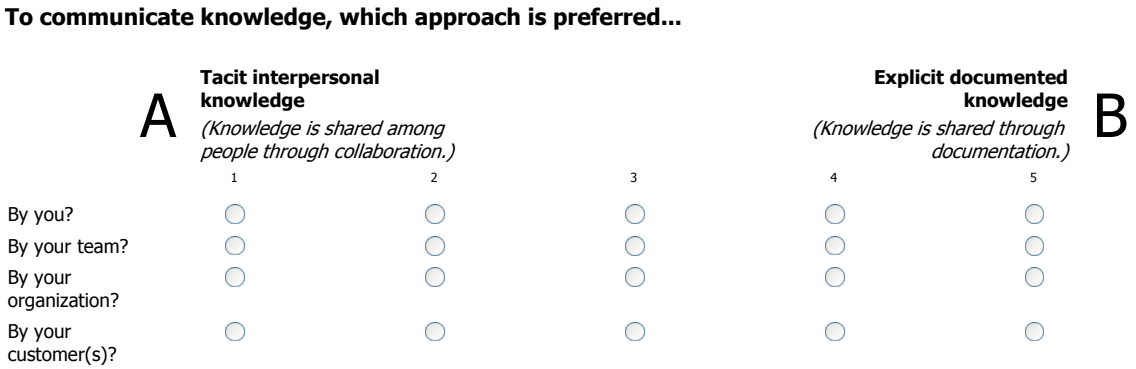


Figure 4.7: The question inferred from the measurement area Knowledge Distribution

4.5 Context Information

Beside collecting information about the respondents attitudes towards different concepts within software development, some context information is collected. This context information is gathered for two main reasons:

1. A required part in determining if the findings can be generalized.
2. In order to answer the research questions:
   - Is there a difference in the degree of agility of IT professionals, and their team, organization and customer?
   - Is there a correlation between IT professionals' characteristics, i.e. their age, job function, experience, education and their agility?
   - Is the context of software projects suited for Agile Software Development?

The needed context information is divided into three groups; *Personal Background*, *Organizational Background*, and *Project Context*. This grouping of the context information is merely a conceptual ordering for a better overview.

4.5.1 Personal Background

Personal background concerns the respondents personal characteristics, i.e. age, job function, experience, and education. This kind of information is needed because the variation in the sample can affect the assumptions drawn from the findings as mentioned in Section 4.1.

The main factor that is relevant with regard to sample characteristics is the variation of the sample. If the sample is homogeneous with respect to the characteristics under study, a small sample can provide a reasonably good estimate and the opposite is the case if the sample is heterogeneous.

Clearly not all kinds of personal background information is relevant. As an example respondents *Shoe Size*, is not seen as very relevant. The importance of a personal background metric should be seen in regards to its possible influence on the subject investigated. For instance, it is not believed that the size of respondents shoes has anything to do with their agility. However, people's age clearly influence the choices they make, their believes and experiences and could thus have more impact on agility.

Personal background information is also gathered in order to answer some of the research questions. These research questions are based on the assumption that some variables, e.g. age and professional experience, could likely be related to the agility of IT Professionals.

Personal Background Metrics

We desire to design a clean and as small a questionnaire as possible. Information about sex and postal code is some of the metrics that are excluded from the final questionnaire. Sex would perhaps be useful in order to determine the variation of the sample, but it is not deemed important when examining peoples agility and thus excluded. Retrieving the postal code of respondents can help establish the geographical relationship of respondents. However, it is decided to ask respondents about their company name instead. Knowing the name of the respondent's company can not directly provide exact geographical information, as large companies are likely to have more than one department across the country. However, knowing the company name is seen as more important than just geographical placement, as it would make it possible to retrieve additional information such as business area, financial situation, company age, etc. through the company web site or annual results.

The following personal background metrics are selected:

- Age
- Educational background
- Years of professional software development experience
- Primary job function(s)

With regards to age, respondents are given the opportunity to state their *year of birth* by using a dropdown option box that range from 1940-1990.

Educational background does not refer to the specific education respondents have, but to the kind of education (Ba., Ma., Ph.D). Possible answers are given in a multiple choice set:

- Self-taught

- Training course(s)
- Bachelor degree
- Master degree
- Ph.D.
- Other further education

As a multiple choice question set the respondents can choose more than one educational background. If respondents feel that their educational background is not on the list, they have the opportunity to specify it using the *other further education* option.

Further more, the respondents are asked to state how many years of professional software development experience they have. Again a dropdown option box is used with a range from 1-30 years and above.

Finally the respondents are asked to state their primary job function(s). The information gathered here should enable the separation of developers from the rest. Agile methods, especially with XP as the flagship, can be seen as developer centric, because they give more responsibility to developers (technical decisions), but also removes some (business decisions). In order to avoid confusion in the question, "job title" being a common term in business today, is more appropriate than "job function". However, today there exist a vast number of different job titles, which will make the effort to find and list them all more cumbersome. Also it is believed that many job titles obscures what people actually do more than bringing clarity to it. The information required is to know what people actually do, e.g. develop, manage, gather requirements, etc. and therefore it is decided to use the term "job function".

Respondents can select between the following job functions:

- Tester (E.g. structured test, unit test, acceptance test, usability test, etc.)
- Programmer
- System Architect
- Manager (E.g. project manager, team leader, etc.)
- Feature Analyst (E.g. gather system requirements, customer contact, etc.)
- Other

4.5.2 Organizational Background

Organization background concerns the characteristics of the respondents organization, i.e. name, size, culture, etc. Hansson et al. [22] concludes that agility not only depends on which value is judged to assess agility but also the characteristics of the organization and the project. Therefor this is an important part of the questionnaire.

A lot of information about the respondent's organization can be gathered, however, not all information is relevant in the perspective of this research project. For instance information about the organization's annual turnover is not relevant information, but the organization's culture is relevant, because it could have an impact on the respondent's agility. Some organizational information can be gathered without the involvement of the respondents, it only depends on which metrics is used to gather the information.

Organizational background information is also gathered in order to answer some of the research questions.

Organizational Background Metrics

The following organizational metrics are gathered:

- Name of the organization
- Organization size
- Development culture
- Used standardized software development method(s) or model(s)?
- Adherence to the software development method(s) or model(s)?

The respondents are asked to state the name of the organization they are currently working for. This information can be used in different ways. It is possible to collect general information about the organization without asking the respondent about them, hence making it a less tedious questionnaire for the respondents. But not all information are possible to collect from an organizations web site, therefore other questions are asked with regard to their development culture and their development method(s) they use in their organization. With the size metric, information about the size of the organization is gathered.

The last two metrics concern the standardized software development methods and model used, and their adherence to them. This information will give an indication of how many organizations say they use a development method or model, and if they actually adhere to the method and model. The information will also indicate if the organizations currently is using agile methods or at least saying they use agile methods.

4.5.3 Project Context

Project context concerns the actual projects the respondents are currently working on, i.e. team, duration, criticality, and the transformation of Boehm and Turners [15] context factors into questions.

A lot of information about each respondent's project context can be gathered, however, not all information is relevant in the perspective of this research project. For instance information about how much money have been spent on the project, is not relevant information.

The project criticality is relevant, as it indicates if the project fits in a agile context, and is one of Boehm and Turners factors.

Project Context Metrics

Due to the reasons found in Section 2.4, the Boehm and Turner agile and plan-driven home-ground model [15] is used to gather project context information. The Boehm and Turner polar chart shown in Figure 4.8 is used.

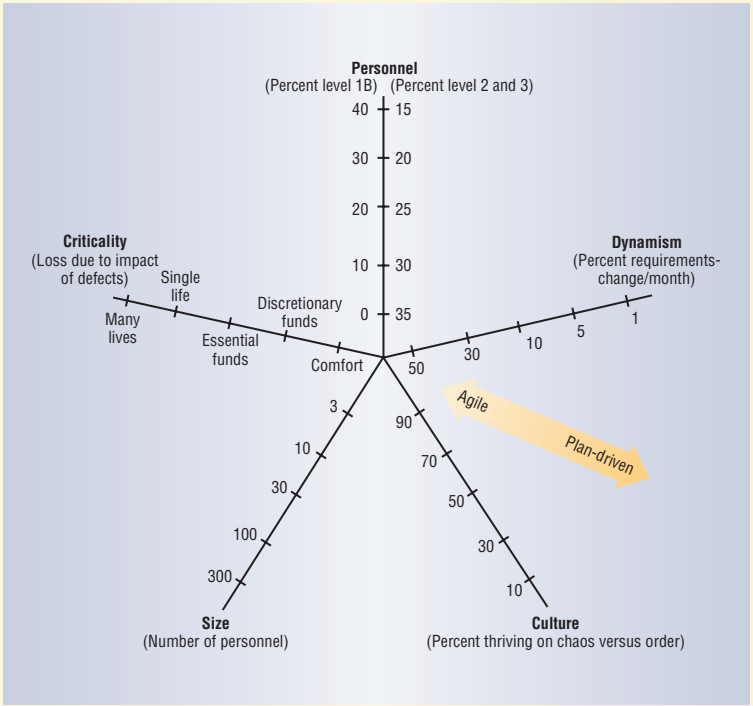


Figure 4.8: Polar chart displaying the five context factors developed by Boehm and Turner [15].

The factors shown in Figure 4.8 are transformed into questions so that project context information can be gathered from the respondents through the questionnaire. There are five factors:

1. Personnel
2. Dynamism
3. Culture
4. Size
5. Criticality

However, only four of the five factors are used; *Dynamism*, *Culture*, *Size*, and *Criticality*. The Boehm and Turner *personnel* factor, is based on Cockburns Three Levels of Software

Understanding, which they have modified to contain 5 levels. This skill level and its influence on agility is described in Section 2.4. This scale is difficult to transform into a question that can be used in a questionnaire. It also requires that the respondent understands the scale and the levels described to be able to state their skill level. Besides that it is a subjective judgment from the respondent where as the other factors are rather objective.

However, some of the personal background metrics collected from the respondents, can be used to establish some measure of skill level, e.g. by looking at the years of experience and educational background. This is not how it is measured in the Cockburns skill scale and should be used with caution.

The transformation of the factors is described in the following subsections.

Dynamism

Dynamism concerns the level of change rate in a project. Agile methods work well with high and low change rates, whereas plan-driven methods works best with low level of change rates [15]. The change rate is related to the percentage of requirement changes per. month.

This factor is transformed without any complications, as the factor is easy to understand and the scale is quite clear and straight froward to implement. The scale goes from 1% - Above 50%. The question is formulated as:

- How large a percentage do your current project requirements change during one month?

This question can be objective, depending on how and if the changing requirements are documented frequently during their project. However, if that is not the case, the answer is rather subjective and can be subject to errors.

Culture

Culture concerns thriving on chaos or on order. The culture reflects that agile methods will succeed better in a culture that thrives on chaos than in one that thrives on order. This question is also rather straight forward to transform, as the values from the scale in the polar chart (Figure 4.8) is used. The culture question is formulated as:

- How would you classify the development culture in your organization?

This question is subjective, because respondents have to evaluate how large a percentage of their team thrives on chaos or order, e.g. 60% chaos and 40% order.

Criticality

Criticality concerns the loss of life due to impacts of defects in a system. This question is also straight forward to transform, and the scale in the polar chart (Figure 4.8) is used. The criticality question is formulated as:

- How would you classify the criticality of the system you are currently developing?

This is a rather objective question, as it should be clear for the respondent which kind of systems they are currently working on, and if lives depend on it or not.

Size

Size concern the number of personnel. Again this factor is straight forward to transform into a question with an understandable scale. The scale that is used is the same as in the polar chart (Figure 4.8), and it goes from 3 - Above 300. The size question is formulated as:

- What is the size of your current project team?

The question is objective. However, the number really depends on whom the respondent includes to be part of project team, i.e. only the developers or all the stakeholders.

Beside the above mentioned factors, a project duration metric is used to gather information about project durations. This is used to see if a broad segment of the projects are covered, hence small, medium and large projects.

5 Analysis

The data collected through the questionnaire need to be processed and analyzed in order to answer the research questions and validate the research. The following sections presents the theory behind analysis and the results and findings of the analysis.

5.1 Theory

Analysis theory gives a introduction to the statistical methods employed to answer the research questions. It also gives a brief description on some of the main statistical concepts and tools.

5.1.1 Statistical Concepts

There exists a number of methods to numerically describe the characteristics of a data set. These different methods are explained in this section along with other statistical concepts that should be clarified.

Steven's Classification of Variable

To distinguish different types of variables from each other a classification system is used. One of the widely accepted classification systems is called the *"Steven's Variable Classification System"* [1]. It consists of four different variable types and determine the kind of analysis that can be performed on them. These are listed bellow [1]:

Nominal is when each variable belongs to one of several distinct categories. E.g. company names, race, religion, etc.

Ordinal variables are also category variables, but there also exists a known order among them. E.g. hardness of minerals, socioeconomic status, etc.

Interval is a variable in which the differences between successive values are always the same. E.g. temperature in degrees (Fahrenheit or Celsius), calendar dates, etc.

Ratio is the same as interval variables but with a natural zero point, meaning that the variables have values below zero. E.g. Height, weight, etc.

As mentioned above, these classifications help determine the method of analysis which is possible. Nominal variables are the most restrictive in that there exists no order among the categories and therefor statistical analysis cannot be performed. The interval and ratio variables are the most statistically useful variables as the mean, median, standard deviation, etc. can be calculated. The ordinal variables are more permitting than nominal, but researchers handle them differently. Some treat them as nominal variables where as others interpret them as interval variables. This of course makes a great deal of difference to the conclusions that can be drawn from the variables [1]. However, it is argued that it is up to the researcher to balance the variable classification, validity of the conclusions, and the usefulness of the analysis method [1].

Mean, Median and Mode

The Mean and Median is used to measure the central tendencies of a data set.

The *mean* of a sample is the sum of the data values divided by the number of observations.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + ... + x_i}{n}$$

The *median* is the middle observation of a set of observations that are arranged in increasing or decreasing order. If the sample size, n, is an odd number, the median is the middle observation. If n is an even number, the median is the average of the two middle observations. The median is located in the $1/2(n+1)$th ordered position.[10]

The *mode* is the most frequently occurring value of a specific variable.

Measuring Data Dispersion

To further describe the data a number of methods are used to measure the spread or dispersion of the data from the mean.

Range is the difference between the largest and smallest observations. This, however, only accounts for two of the data values. A measure that takes all the data values into account is needed. This measure would average the total distance between each observation and the mean. Since this number would be negative for values smaller than the mean it is squared because a distance cannot be negative $(x_i - \bar{x})^2$. The average of the sum of squared terms is called the variance.[10]

The *sample variance* s^2 is defined by:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

To get the data in its original measurement units the *standard deviation* is used. Since the variance squared the values to account for negative distances the standard deviation takes the square root of the variation.

$$s = \sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Measuring Relationship Between Variables

When graphically describing the relationship between two variables a scatter plot is used. This is done by plotting each pair of observed values from the data set into a co-ordinate system where the variables are plotted on the x and y axis respectively.

To measure the relationship numerically *covariance* (Cov) and *correlations* are used.[10] Covariance is a measure of the linear relationship between two variables. A positive value indicates an increasing linear relationship, and a negative value indicates a decreasing linear relationship. The sample covariance is defined as:

$$Cov(x,y) = s_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

This provides the direction of the linear relationship between the two variables x and y. To also measure the strength of the relationship the correlation coefficient (r) is used.

$$r = \frac{Cov(x,y)}{s_x s_y}$$

The correlation coefficient ranges from -1 to +1 and the closer r is to +1, the closer the data points are to an increasing straight line, which indicates a positive linear relationship. The closer r is to -1, the closer the data points are to a decreasing straight line, which indicates a negative linear relationship. If $r = 0$ there exist no linear relationship between the variables.[10]

Variable Normality

Variable normality indicates something about how the data is distributed for a single variable. A *standard normal distribution* is when the majority of data falls within the center

values and then steadily diminishes towards the edges with a mean of zero and variance of one. The best way to describe this, is to use histograms. Figure 5.1 show a histogram of four normal distributions. The green line is the standard normal distribution. Figure 5.1 also illustrates other normal distributions that are skewed to the left of the center changing the mean, and varying in hight changing the variance. Samples from real world cases with large data sets often approximate some form of normal distribution.[10]

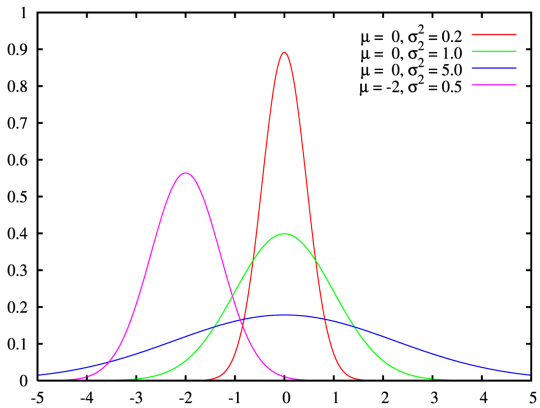


Figure 5.1: Histogram of several normal distributions [40]

Missing Values

Missing values can come in two forms; *unit nonresponse* and *item nonresponse*. Unit nonresponse is used in estimating the response rate of the survey and denotes the sample respondents that do not answer the questionnaire. This kind of nonresponse cannot be seen in the data set itself as no variables are available from the respondent, but it can have a significant impact on the power of the conclusions drawn from the data.

Item nonresponse is when a respondent does not answer one or more of the questions in the questionnaire. This manifests itself as missing values in the data set. Most statistical analysis software packages simply delete the cases where one or more variable values are missing. This can, however, result in an unacceptably low number of respondents. If the missing values are randomly distributed and limited, another approach is via *imputation* (substitution) of the missing values. This is an area where a great deal of statistical research has been done and there exits many methods for doing the imputation. Each of these methods have their own pros and cons concerning data validity and processing complexity but these will not be discussed further in this report as they are not used in this project.[1]

Outliers

Outliers are extreme cases in the data set that are inconsistent with the rest of the data. It is difficult to judge what constitutes an outlier but in larger data sets it can be seen as single cases that deviate excessively from the rest. An outlier can have overly significant influence over statistical measures such as the mean, standard deviation, covariance and correlation. This is why outliers should be identified and removed before analysis. To identify outliers

scatter plots can be used to represent the data graphically. Formal tests for outliers also exist, but they usually assume variable normality and are quite sensitive to non-normality. So these should only be used when the assumption of variable normality is reasonable. To test if outliers make a significant difference in the analysis, two test can be run. One with the outliers in the set and one without.

5.1.2 The Analysis Method

The analysis method that is used to explore the collected data and assist in answering our research questions is called *"Factor Analysis"*.

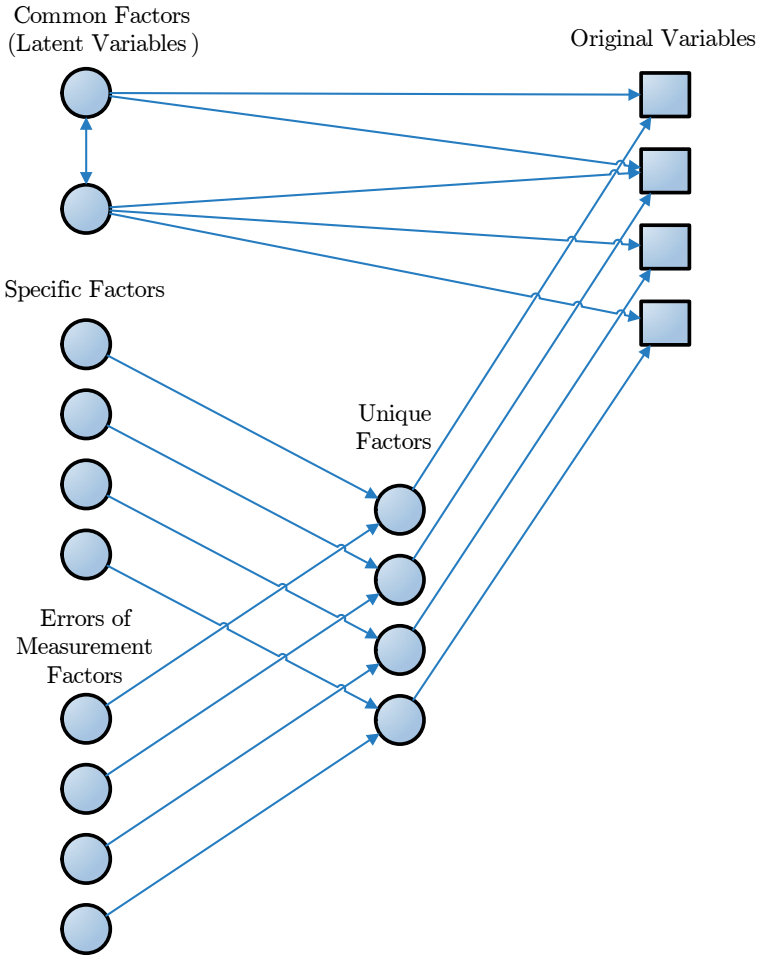


Figure 5.2: A schematic for Factor analysis theory (The factor model).

It is an exploratory method for looking at dependencies among variables. It falls under the category of data reduction analysis methods, because it sets out to describe the majority of the data by creating fewer new variables called *common factors*. As it is an exploratory method the results from the analysis is subject to interpretation by the researcher and should therefore not be used as a basis for new statistical theory. Nevertheless, factor analysis is

very useful when exploring possible correlations between variables and helps the researcher gain insight into the data.[1][7]

Figure 5.2 shows two common factors that have been extracted from four original variables. These original variables hold the data collected from respondents and should ideally all point to or interact with very few common factors, which in turn should be known in advance. In factor analysis not all information is explained by a set of common factors, there are also unique factors to every single original variable, as illustrated in Figure 5.2. These unique factors can be a subset of factors only working on one of the original variables, and an error of measurement factor. The error of measurement factor is not an unobserved characteristic of the individual respondent but an unsystematic transient event[13].

Figure 5.2 is a graphical representation of the mathematical factor model, which is introduced later in this chapter.

Factor analysis consists of two stages called *"Initial Factor Extraction"* and *"Factor Rotation"*. Each of these stages can be performed by different methods and some of these is addressed in the following sections. Factor analysis is often mentioned in correlation with *"Principle Component (PC)"* analysis, which also is one of the methods used in the Initial Extraction stage. PC analysis is a data reduction analysis method in its own right, but the math behind the calculations in the factor and PC analysis methods differ somewhat. This is why the sections bellow start by introducing PC analysis and then discusses Factor analysis.

5.1.3 Principle Component Analysis

PC can be described as a method of transforming the original variables into new, uncorrelated variables, which is called the *Principle Components*. These principle components are linear combinations of the original variables. Each of the new variables conveys a measurable amount of information on the correlation between the original variables, in the form of the new variable's variance. These are then arranged in order of decreasing variance so the first principle component is the most informative variable and the last is the least informative.

As an example, two variables X_1 and X_2 are analyzed. Each variable contains a random sample of N observations. First of, to ease interpretation, the sample mean is subtracted from each observation.

$$x_1 = X_1 - \bar{X}_1$$
$$x_2 = X_2 - \bar{X}_2$$

This makes the mean of x_1 and x_2 zero, but does not alter the sample variances S_1^2 and S_2^2 or correlation r. Now two new variables C_1 and C_2 are crated from linear combinations of x_1 and x_2. These are the principle components and written as

$$C_1 = a_{11}x_1 + a_{12}x_2$$
$$C_2 = a_{21}x_1 + a_{22}x_2$$

Note that for any set of values of the coefficients a_{11}, a_{12}, a_{21} and a_{22} the N observed x_1 and x_2 can be introduced and thereby obtain N values of C_1 and C_2. The means and variances

of these are

$$\begin{aligned} \text{mean}C_1 &= \text{mean}C_2 = 0 \\ \text{Var}C_1 &= a_{11}^2 S_1^2 + a_{12}^2 S_2^2 + 2a_{11}a_{12}rS_1S_2 \\ \text{Var}C_2 &= a_{21}^2 S_2^2 + a_{22}^2 S_2^2 + 2a_{21}a_{22}rS_1S_2 \end{aligned}$$

Where $S_i^2 = \text{Var}X_i$

Now the coefficients are chosen within the confines of these three rules:

1. Var C_1 is as large as possible.
2. The N values of C_1 and C_2 are uncorrelated.
3. $a_{11} + a_{12} = a_{21} + a_{22} = 1$

In affect C_1 and C_2 are rotations of the original x_1 and x_2 axes and the N values of C_1 will have the largest variance as per rule number one. These variances are also known as the principle components *eigenvalues* of the covariance matrix of X_1 and X_2. Also the eigenvalues of all principle components add up to the total variance of the original variables. All this means that the new principle components are rotated in such a way that the variance of the original variables is distributed so the first principle components describes as much of the information as possible.

Most researchers prefer to standardize the variables prior to analysis because it compensates for the unit of measure of the variables and eases interpretation. This standardization is achieved by dividing each variable by its sample standard deviation, which in turn is equivalent to analyzing the *correlation matrix* instead of the *covariance matrix*. For the correlation matrix the total variance is the number of variables analyzed (P), and the proportion explained by each principal component is the corresponding eigenvalue divided by P.

Every PC analysis on P variables produces P principle components, but since the object is to reduce the number of variables only a subset is chosen. This subset is based on the amount of the total variance explained by the chosen principle components. A rule of thumb is that all principle components with eigenvalues larger than 1 is selected. However, analysis of *scree diagrams* and total percent of variance explained is also used to make the decision of how many principle components to select.

5.1.4 Factor Analysis

In PC analysis the major objective is to find a number of principle components that explain as much of the total variance as possible. However, factor analysis is mainly used to explain the interrelationships among the original variables. Ideally, the number of expected factors is known in advance and they should be easily understandable and relate the information contained in the original variables.

As an example, the variables $X_1, X_2.....X_P$ are used. Again these variables are standardized $x_i = (X_i - \bar{X}_i)/S_i$, so that their variances are equal to one and their covariances are correlation coefficients. Now it is the objective of factor analysis to represent each of these variables as a linear combination of a smaller set of *common factors* plus a factor unique to each of the response variables. The common factors are unobserved characteristics that have an impact on the correlation between variables. Whereas the unique factor can consist of specific factors that do not influence variable correlation, or errors of measurement factors[13].

$$\begin{aligned} x_1 &= l_{11}F_1 + l_{12}F_2 + ... + l_{1m}F_m + e_1 \\ x_2 &= l_{21}F_1 + l_{22}F_2 + ... + l_{2m}F_m + e_2 \\ &\vdots \\ x_P &= l_{P1}F_1 + l_{P2}F_2 + ... + l_{Pm}F_m + e_P \end{aligned}$$

Where the following assumptions are made:

1. m is the number of common factors (typically much smaller than P)
2. $F_1, F_2, ..., F_m$ are the common factors.
3. l_{ij} is the coefficient of F_j in the linear combination describing x_i
4. $e_1, e_2, ..., e_P$ are unique factors, each relating to one of the original variables.

The above equation and the four assumptions constitute the *factor model*. Where each of the response variables are composed of a common factor and its own unique factor. When the factor model breaks the response variable x_i into two parts, it also breaks the variance of x_i into two parts:

1. The *communality*, i.e., the part of the variance that is due to the common factors.
2. The *specificity*, i.e., the part of the variance that is due to the unique factor e_i.

If the communality of x_i is denoted by h_i^2 and the specificity by u_i^2, the variance of x_i is Var $x_i = 1 = h_i^2 + u_i^2$. In other words the variance is the sum of the communality and the specificity. Factor analysis is concerned with finding estimates of the factor loadings l_{ij} and the communalities h_i^2. One of the ways to solve this equation is through the use of PC analysis and this is called the *initial factor extraction* as mentioned earlier.

Initial Factor Extraction Using Principle Components

To use the PC analysis for the factor extraction the principle component model has to be modified to express each variable x_i in terms of F_j's. To do this first recall the relationship

between the variable x_i and the principal components C_j.

$$\begin{aligned} C_1 &= a_{11}x_1 + a_{12}x_2 + ... + a_{1P}x_P \\ C_2 &= a_{21}x_1 + a_{22}x_2 + ... + a_{2P}x_P \\ &\vdots \\ C_P &= a_{P1}x_1 + a_{P2}x_2 + ... + a_{PP}x_P \end{aligned}$$

This set of equations can be inverted to express the x_i's as functions of the C_j's.

$$\begin{aligned} x_1 &= a_{11}C_1 + a_{21}C_2 + ... + a_{P1}C_P \\ x_2 &= a_{12}C_1 + a_{22}C_2 + ... + a_{P2}C_P \\ &\vdots \\ x_P &= a_{1P}C_1 + a_{2P}C_2 + ... + a_{PP}C_P \end{aligned}$$

Now the first set of equations has been transposed so the rows of the first set has become the columns of the second set of equations. Since factors are assumed to be standardized we divide the principle component by its standard deviation, so $F_j = C_j/(\text{Var}C_j)^{1/2}$ and thereby $C_j = F_j(\text{Var}C_j)^{1/2}$. This gives the ith equation

$$x_1 = a_{1i}F_1(\text{Var}C_1)^{1/2} + a_{2i}F_2(\text{Var}C_2)^{1/2} + ... + a_{Pi}F_P(\text{Var}C_P)^{1/2}$$

This is then modified in the following two ways:

1. The notation $l_{ji} = a_{ji}(\text{Var}C_j)^{1/2}$ for the first m components is used.
2. The last $P - m$ terms are combined and denoted by e^i.
   $e^i = a_{m+1,i}F_{m+1}(\text{Var}C_{m+1})^{1/2} + ... + a_{Pi}F_P(\text{Var}C_P)^{1/2}$

This results in each variable x_i being expressed in the form

$$x_i = l_{i1}F_1 + l_{i2}F_2 + ... + l_{im}F_m + e_i$$

For $i = 1$ to P. Now the principle component model has been transformed to produce the factor model. It should also be noted that when the variables are standardized the factor loadings l_{ij} are the same as the correlations between x_i and F_j in correlation matrix also called the pattern matrix.

As explained in the section about PC analysis this extraction tries to maximize the variance explained by the first factors and thereby explain the majority of the data. Other factor extraction methods exist that take a different approach. One such method is *principal axis factoring* that instead of maximizing the variance explained, maximizes the communalities. However, this method will not be described further in this report.

Factor Rotation Using Verimax

The idea behind factor analysis is to find common factors that can easily be explained. However, it is not always possible to interpret the resulting factors from the initial extraction, so a rotation is performed. When rotating the original factors new factors called the *rotated factors* are created. These rotated factors are selected so that some loadings are very large (near ± 1) and the remaining loadings are very small (near zero). Also factor loadings that are high in one factor is minimized in the other factors. This makes the resulting factors easier to interpret and use in further analysis.

Verimax rotation is an approach that assumes that the rotated factors are uncorrelated and thus the axis of these factors are orthogonal (perpendicular) to each other. Other rotation methods exist that relax this requirement, but these are not discussed any further.

5.1.5 Cronbach's Alpha

Cronbach's Alpha is a method of measuring the reliability of a statistical research instrument, developed by Cronbach in 1951. This method gives the researcher a way to testing if the variables used to measure a common factor or latent variable is reliable. Cronbach's alpha is defined as:

$$\alpha = \frac{N \cdot \bar{r}}{1 + (N-1) \cdot \bar{r}}$$

Where N is the number of variables and $\bar{r}$ is the average of all correlation coefficients between the variables.

Alpha can be considered an unbiased estimator of reliability as long as the variables covariances are equal, which in turn means that they all have one common factor in a factor analysis. In most research on real world data this is, however, not the case when data is collected through questionnaires and is subject to interpretations. This means that it is unlikely that the data follows a linear regression on one common factor and therefor Alpha can be considered a lower bound estimator of reliability.

Alpha can assume values from negative infinity to 1. As a rule of thumb researchers hesitate to use instruments that have an Alpha score of less than 0.70. However, the appropriate degree of reliability depends on the use of the instrument.

It is important to note that the measure of reliability of an instrument does not say anything about the validity. Alpha in a sense measures the degree of correlation between the variables in the instrument and not if what they measure is the correct thing.

5.2 Validity and Reliability

Establishing the validity of our findings and the reliability of our instrumentation is very important. In Section 4.2 and Section 4.3 the concepts of validity and reliability is explained.

The importance lies in, that we as researchers produce valid conclusions based on our findings as mentioned in Section 4.2. The results are based on 142 respondents that completed the questionnaire and answered all questions, out of 304 respondents.

5.2.1 Sample Characteristics

With regard to sample characteristics as described in Section 4.1 it is important to gather respondents who reflect the characteristics of the target population. Besides that we would like to examine if the results are generalizable. Sample characteristics, such as the respondents' age, educational background, professional experience, organization and organization size is used to establish reliability, validity and generalizability. The data gathered about the respondents are also used when answering the research questions. This will be treated later in this chapter. In the following subsections each of the results about the sample will be shown and discussed.

Age

To see if a broad spectrum of the population with regard to age is covered, respondents year of birth is gathered. 142 respondents answered the question: *Year of birth?* and the results are shown in Figure 5.3. The ages are grouped into intervals of five years, i.e. from 21-25, 26-30, etc.

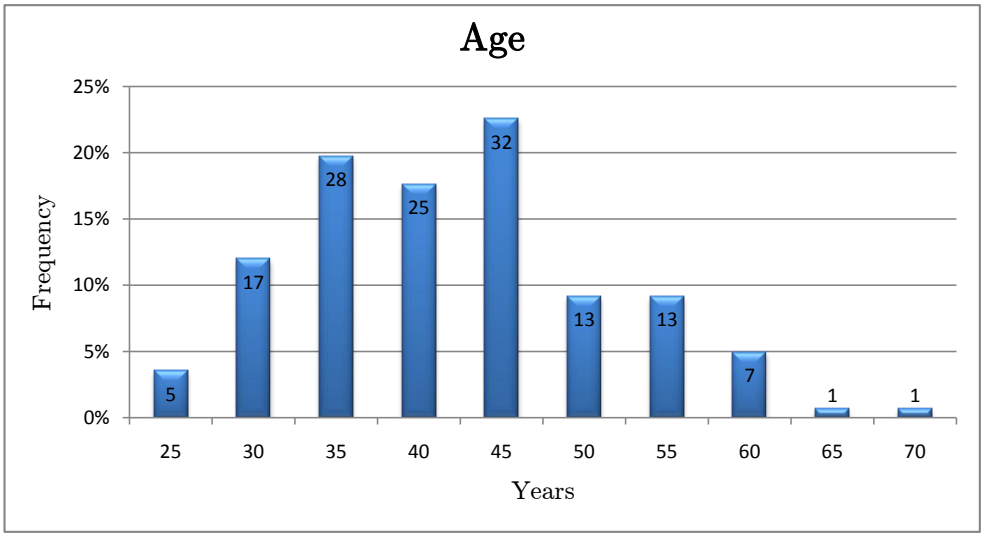


Figure 5.3: The distribution of the respondent's age.

The range is 44 years (22-66 years). The mean and median is 40 years and there is a multiple mode, 31 and 43 years. Figure 5.3 illustrates how age is distributed among the respondents. The figure shows that there is a broad coverage with regard to age.

Educational Background

To see how many of the respondents have an academical education or are self-though, they were asked to specify their educational background, i.e. Bs., Ms, Ph.D., etc. 142 respondents answered the question: *What is your educational background?* and the results are shown in Figure 5.4.

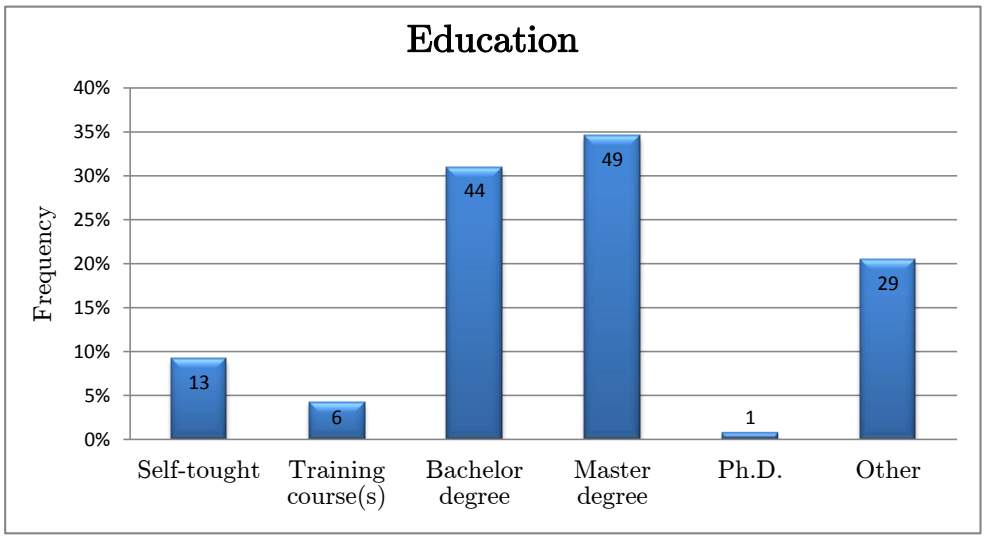


Figure 5.4: Showing the educational background of the respondents.

Figure 5.4 shows that the majority of the respondents 35% (49) have a Masters degree whereas 31% has a Bachelor degree. Additionally, it shows that 9% (13) are self-taught, nearly 4% (6) have been taking training courses, and less than 1% (1) have a Ph.D. Further it is seen that 20% (29) responded that they have "further education" not mentioned in the list of choices. 52% of those said that they had a Computer Science educational background. It was observed that some of the respondents misinterpreted the last option "other further education" as more than one had selected an education from the option list and also stated that they had other further education. However, this option was intended to be used by the respondents that do not have an education that are listed. This means that the reliability of the questions is not strong, as the answers differ from respondent to respondent. Hence, it is interpreted differently by the respondents. But if we abstract from the fact that some respondents did misinterpret the last option, the results show that the respondents are from a wide range of educational backgrounds.

Experience

To see how many years of professional experience the respondents have with software development, the respondents were asked to state how many years of professional software development experience they had. 142 respondents answered the question: *Years of professional software development experience?* and the results can be seen in Figure 5.5. The years of experience have been grouped into intervals of five years.

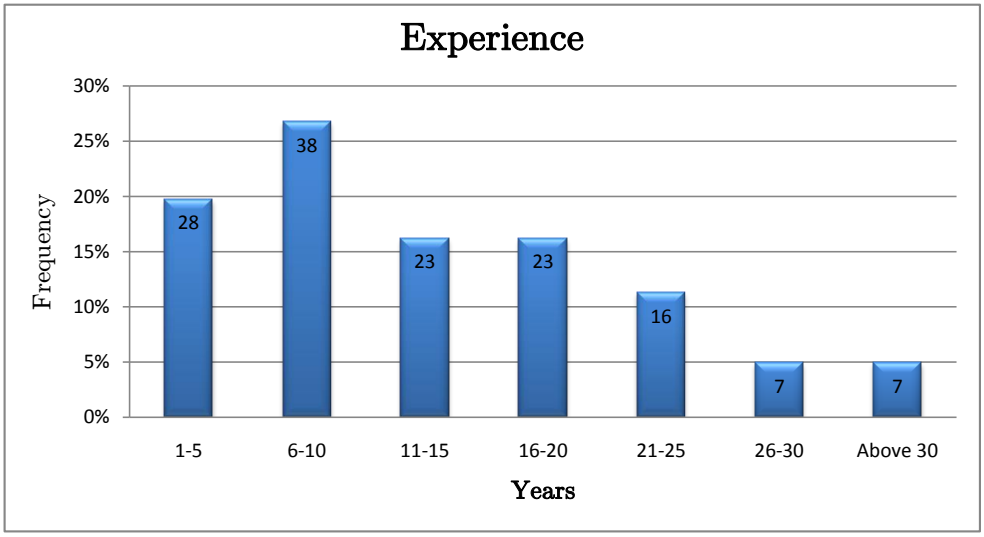


Figure 5.5: Showing the years of experience the respondents have with software development.

Figure 5.5 shows that, the years of professional experience span from 1 to above 30 years. The mean is 13 and the median is 12, whereas the mode is 10. The figure shows that we have respondents from all levels of experience. This indicates that from those that participated, we have covered a broad range of the respondent with different levels of experiences within software development. There are no actual peaks in years of experience, besides in the interval from 6-10 years, where slightly more than 25% of the respondents are.

Job Function

To assess if respondent with different job functions are represented in the data, respondents were required to state which primary job function they have. 142 respondents answered the question: *What is your primary job function(s)?* This question is a multiple-choice question, which means that respondents could state more than one *job function*.

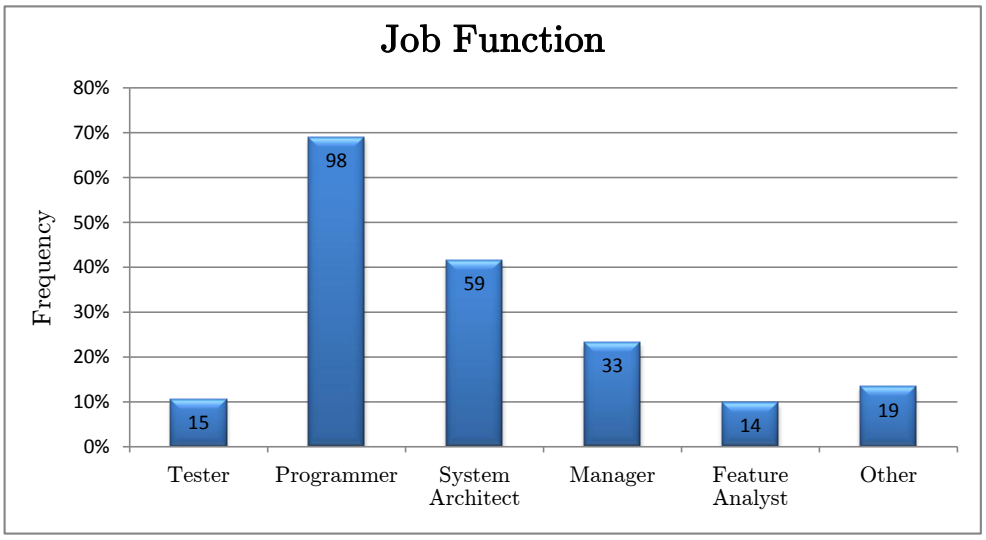


Figure 5.6: Showing the respondents primarily job function(s).

Figure 5.6 shows that most of the respondents 69% (98) of the respondents are programmers. 42% (59) respondents are system architects, 23% (33) are managers, 11% (15) testers, 10% (14) feature analyst and 13% (19) responded that they had other job functions than those listed. The other job functions, besides those that were listed, were: *teacher, professor, consultant, configuration manager, test manager, operations manager, product manager, systems developer, process improvement, server admin, and network surveillance.* The results show that there are different respondents with different job functions that participated. The intent was to cover a broad part of job functions, even though most of the respondents are programmers. Which we assume reflects the industry very well, because there are always more programmers than e.g. managers, feature analyst, etc.

Organization and Size

To see if we have respondents from different organizations, the respondents were asked to state the name of their organization. Not all respondents answered this question, but those who did, come from 85 different organizations here in Denmark. The size of the organizations also varies. The respondents were asked to state how large their organization is by indicating how many employees the organization has. This question was asked to see if we have respondent from different sizes of organization and not only respondents from e.g. small organizations. 142 respondents answered the question: *How large is your organization approximately?* and the results are shown in Figure 5.7.

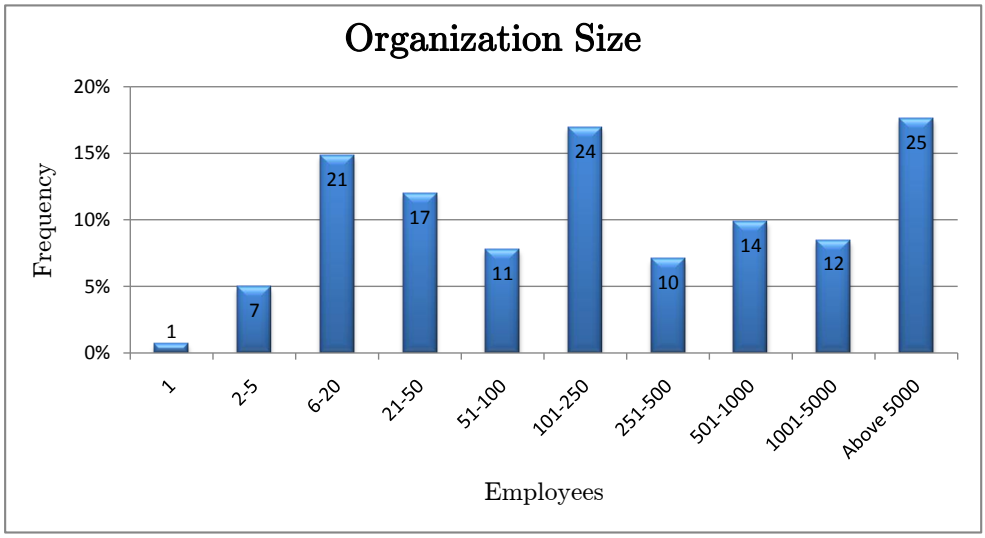


Figure 5.7: The sizes of the organizations that the respondents work for.

Figure 5.7 shows that respondents come from organizations ranging from small to very large organization. The figure shows that there are almost an equal number of respondents from small (6-20 employees), medium (101-250 employees), and very large (above 5000 employees) organizations. However, most of the respondents come from organizations that are very large.

To sum up, it is possible to say that our results indicate that the sample representation does not have any significant holes. However, it is not possible to generalize our data as

the sample can not be confirmed as representative in relation to the target population. The data can still be used to answer the research questions.

5.3 Adjustments of the Measurement Instrument

Before the collected data can be analyzed the measurement instrument variables has to be tested for internal reliability. In our case the measurement instrument is comprised of the seven bipolar preferred questions established in Section 4.4. The reliability test is done using the Cronbach's Alpha reliability indicator introduced in Section 5.1. Furthermore, to measure respondents' agility a benchmark is established that distinguish between agile and non-agile respondents. The final adjustment is a weighting of the variables that comprise the measurement instrument.

Danish Agile User Group and Main Survey Test Sets

The survey was conducted on a primary data collection group and on a secondary agility test group, DAUG. The primary group consisted of IT professionals and provided the data for answering the research questions. The respondents were gathered as explained in Section 4.1. The secondary group of respondents were members of DAUG and acted as a testbed for the agility measurement instrument. The two groups both answered the same questionnaire but the data collection was independent of each other. To isolate the two groups from each, other steps were taken to minimize the chance of one respondent answering more than one questionnaire. First of, the data collection from the two groups was done sequentially, where the main survey was conducted first and there after the DAUG survey. Furthermore the respondents from the main survey were asked if they were a member of DAUG and then excluded from the DAUG survey.

The goal of the DAUG survey was to establish an agility benchmark on our measurement instrument and test the internal reliability. This would be achieved by collecting data from the DAUG members, which were known to be agile and then relate the agility score of these respondents to that of the respondents from the main survey.

5.3.1 Cronbach's Alpha Internal Reliability

One indicator of internal reliability is Cronbach's Alpha, which is explained in detail in Section 5.1. In short it provides a measurement of how strong the correlation between the variables in the instrument is. This can be used as an indicator as to how well the instrument measures a single factor, in our case agility. However, Cronbach's Alpha can not be used as an indicator for validity in that it does not take into account what the instrument measures but only how well it measure something. The process of arguing that the instrument indeed measures agility is done later in this section.

If Variable Deleted Test

When using Cronbach's Alpha to test a measurement instrument the Alpha value is often calculated on all the variables first to establish an overall measurement. Then the Alpha value is calculated when each variable, in turn, is removed, which gives a table of "if variable deleted" Alpha values. This test is used to indicate if any of the instrument variables are pulling in the wrong direction compared to the overall Alpha.

Both the main and DAUG survey were tested with the "if variable deleted" approach. Table 5.1 shows the overall Alpha values of the two survey groups and the "if deleted" values for each of the seven variables. Remembering that Alpha is an indicator for the degree of correlation between the variables, there exist somewhat of a difference between the two surveys. This can be caused by the fact that considerably fewer respondents participated in the DAUG survey; only 27 compared with 142. I can also be caused by the DAUG respondents being more familiar with general and agile software development methods and thereby sharing a common view on the concepts involved in the questions. If this is the reason it would indicate that the instrument indeed measures agility since agility is the common factor shared by the DAUG respondents and not the main survey respondents.

Bipolar Preferred Questions (Variables)	**Main Survey Cronbach's Alpha**	**DAUG Survey Cronbach's Alpha**
1. Organizational Structure	0,604	0,801
2. Requirements Handling	0,629	0,818
3. Project Progress Control	0,571	0,827
4. Software Development Model	0,601	0,825
5. Project Success	0,615	0,819
6. Design	0,682	0,866
7. Knowledge Distribution	0,613	0,839
Overall Cronbach's Alpha	0,654	0,849

Table 5.1: If deleted table of Cronbach's Alpha values for Main and DAUG surveys.

The reliability of the measurement instrument is given by the value of the Cronbach's Alpha and as a rule of thumb the value should preferably be above 0,7 for there to exist sufficient correlation between the variables to consider the instrument useful. The main survey only has an Alpha of 0,654 indicating that the correlation is below the recommended level, however, the Alpha of the DAUG survey is 0,849 and well above the recommendation. This could indicate that the instrument is working and measuring agility, but that respondents with no common background in agility, disagree more than those with agile backgrounds.

The "if deleted" test show that all except one question contribute to the reliability of the instrument. The only variable that increase the Alpha value if deleted is variable 6. Design. This is the case in both the main and DAUG surveys, which indicates that the question disrupts the correlation between the variables. The design question centers around the agile approach of simple design as opposed to comprehensive upfront design. The Cronbach's Alpha test indicates that the design measurement area is a poor variable for measuring

agility because the people who are otherwise regarded as agile by the instrument prefer, comprehensive design and viceversa. For this reason the design variable is excluded from the analysis and is not used to measure agility. This gives the main survey a Cronbach's Alpha of 0,682 just bellow the recommended 0,7 and the DAUG survey a value of 0,866, which signify that the measurement instrument has a good internal reliability.

5.3.2 Agility Benchmark

Our instrument is constructed to measure agility, as mentioned in Chapter 4, on a scale of 1 to 5, where 1 is agile and 5 is not agile. But what is the cutoff point where respondents go from being agile to non-agile and is there a middle ground between the two sides. The logical choice as cutoff point for the scale would be 3 as it represents the middle of the scale. However, this choice would be based on the assumption that respondents interpret the scale in the same way and observe 3 as the dividing value. Also this choice would not provide a middle ground and only divide respondents in agile and non-agile. Another approach is to establish a benchmark for agility and thereby calibrate the scale. The benchmark is established using the DAUG survey, which is comprised of agile respondents.

Before the DAUG data could be used it had to be edited. Only the respondents who stated in the questionnaire that they used agile methods are included. This was necessary because some of the respondents stated that they used non-agile methods and homemade methods, which could not be determined as agile, and these respondents could influence the agility rating. This elimination meant that the 27 DAUG respondents was reduced to 13 which all used agile methods.

Each of these respondents was then plotted on a scatter plot shown in Figure 5.8. The respondents are plotted according to their agility score on the vertical axis and their variance on the horizontal axis.

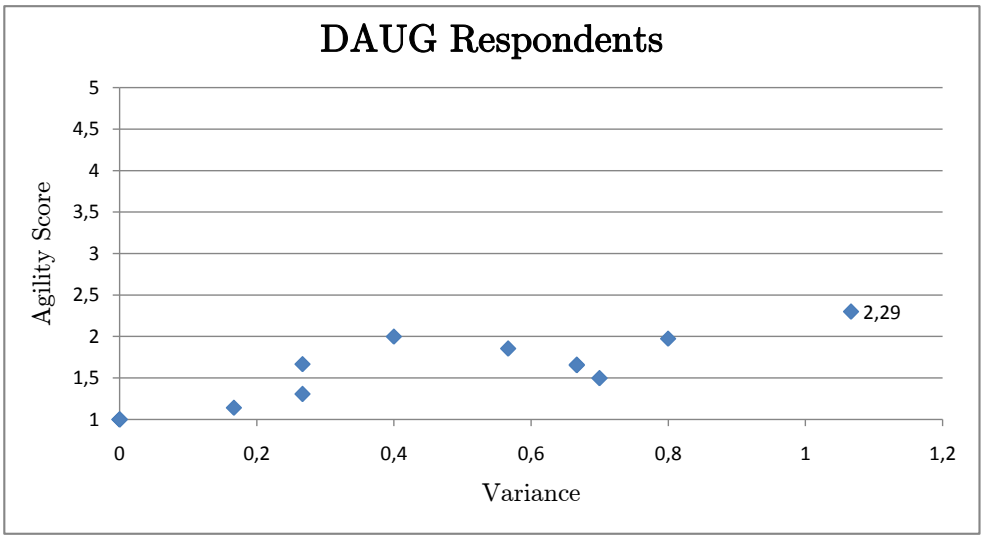


Figure 5.8: Scatter plot of DAUG respondents.

The agility score on the vertical axis is calculated as a weighted average of the values on

each of the six questions. This means that the agility score of each respondent is the mean of the respondent's answers to the six bipolar preferred questions where each questions have been multiplied by a weight. This weight is explained later in this section. The equation for calculating the agility score is shown in Equation 5.1.

$$\text{Agility Score} = \frac{w_1Q_1 + w_2Q_2 + w_3Q_3 + w_4Q_4 + w_5Q_5 + w_6Q_6}{\sum_{i=1}^{6} w_i} \tag{5.1}$$

The horizontal axis in Figure 5.8 shows the variance of each respondent. This is a measure of how varied the respondent's answers are. So while a respondent alternating the answers between 1 and 5, and one consistently answering 3 would both get a mean of 3, they would not get the same variance. This means that we can measure the consistency of the answers and thereby establish if the respondent is conflicted or confused in relation to agility.

From Figure 5.8 we can also see that all respondents are in the lower half of the scale. This is expected as it is assumed that all the respondents are agile practitioners and should therefore also prefer the agile concepts. If the assumption, that all these respondents are agile is true, the cutoff line for agility can be set at the point of the highest agility score. In this case the highest agility score is 2.29 and it would then be safe to assume that all respondents scoring lower than 2,3 can be considered agile. Now that the benchmark for agility has been established there also has to be a cutoff line for non-agile respondents. However, since the survey has not been conducted on an exclusively non-agile group, this benchmark has to be set in another way. As reasoned before the logical choice would be 3, which divided the scale into three intervals:

Agile	**Middle**	**Non-agile**
[1-2,3]	]2,3-3[	[3-5]

Table 5.2: The three intervals of the agility scale.

5.3.3 Variable Weights

The variable weights mentioned earlier to calculated the agility score was used to further increase the accuracy of the measurement instrument. By adding weights to each of the questions they could differentiate between each other. This would give a sort of Thurstone like scale.

The method for creating these weights, is the principle component analysis, which is described in Section 5.1.

Principle Component Analysis

Principle component analysis attempts to describe as much of the information between correlated variables as possible in as few new variables as possible. The new variables or principle components are linear combinations of the original variables and it is this property we take advantage of to create the weights.

When all the elements of the correlation matrix is positive the first principle component is what is known as a size variable and is in essence a weighted average of the factor the component describes [7]. In our case the measurement instrument measures agility and the variables correlate in regard to this factor, so the first principle component is a weighted average of the "size" of agility. This manifests itself in that the component has hight loadings on all 6 variables in the instrument [7].

Table 5.3 shows the loadings for the first principle component. Given that these are weighted averages of the level of agility they indicate how much the correlation between the variables are based on the individual variables. This means that the loadings from the first principle component can be used as weights for each question in the agility score.

Bipolar Preferred Questions (Variables)	**First Principle Component**
1. Organizational Structure	0,679
2. Requirements Handling	0,562
3. Project Progress Control	0,730
4. Software Development Model	0,608
5. Project Success	0,605
6. Knowledge Distribution	0,547

Table 5.3: The first principle size component's loadings on all 6 questions.

5.4 Answering the Research Questions

This section answers the research questions using data collected from the main survey and according to the adjustments that have been done in the previous section. This means that only six of the original seven bipolar preferred questions are used. Also the agility benchmark from Table 5.2 are used and the variable weight from Table 5.3 are used in Equation 5.1 to calculate the agility score.

5.4.1 Agility of IT Professionals

The first research question states the following:

How Agile are IT professionals?

This is answered through the use of the main survey data and the 6 bipolar preferred questions from the questionnaire. Figure 5.9 shows a scatter plot of the main survey respondents' agility score and variance. The agility score is calculated from Equation 5.1 by inserting the values from their personal preferences and the weights from the principle component analysis. The variance is displayed on the horizontal axis and indicates how conflicting the answers to the 6 questions are. Figure 5.9 also shows two red horizontal and one vertical lines. The horizontal lines denotes the agile and non-agile benchmark lines established in the previous section. The vertical line denotes the cutoff line where the answers conflict so much that the respondent can be categorizes as conflicted or confused. This categorization is of course not clear-cut because there could exist a multitude of reasons why the answers vary a great deal. However, when a respondent has a variance of more than 1,5 it clearly impacts the level of agility regardless of the agility score and this is therefore used as a delimiter between conflicted and non-conflicted respondents.

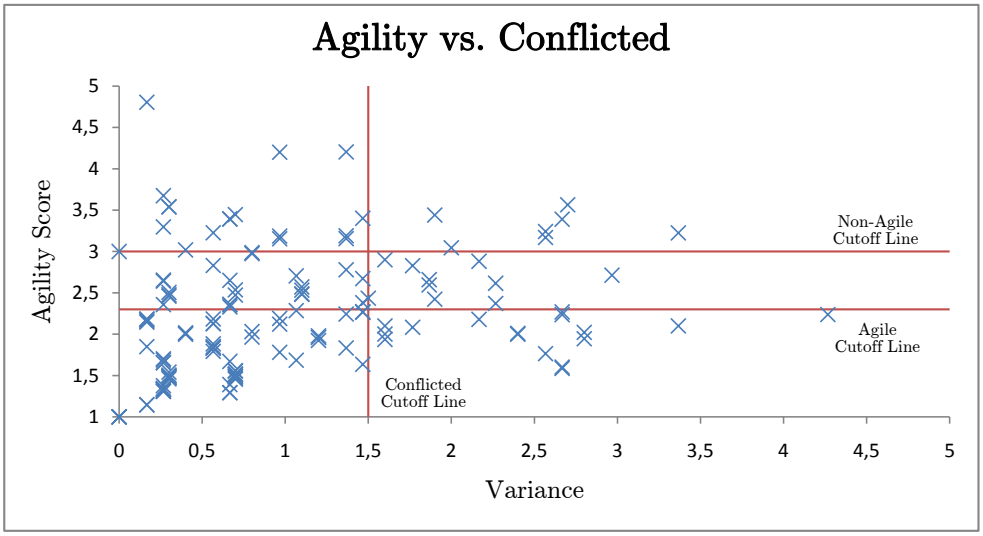


Figure 5.9: Scatter plot of the Main survey's respondents.

The scatter plot illustrates that most respondents fall below the non-agile benchmark and is therefore either in the middle or agile part of the chart.

Figure 5.10 shows this more clearly as each of the bars in the chart represent agile, middle and non-agile respectively. This graph shows that almost 48% of the respondents are agile and an additional 11% are agile but categorized as conflicted. This stands in clear contrast to the middle and non-agile categories that only have 16% and 12% of the respondents respectively. The numbers on each column is the number of respondents the column represents.

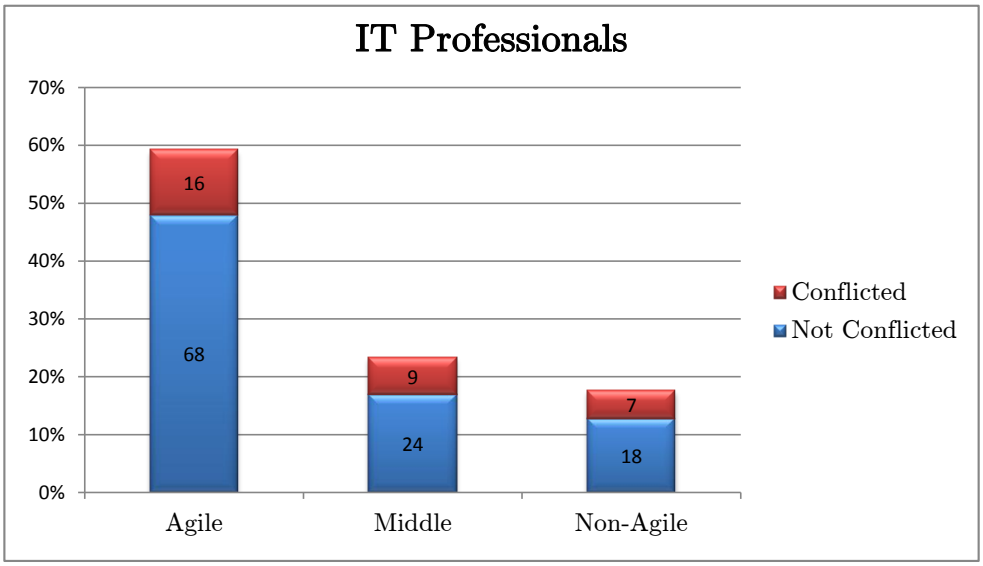


Figure 5.10: Column graph of the agility of IT Professionals.

Figure 5.10 clearly indicates that the majority of respondents are agile but also shows that a considerable percentage of respondents give conflicting answers between the individual questions. This may be caused by a lack of knowledge on agile and general software development, since this this confliction was much less prevalent among the DAUG respondents.

5.4.2 Correlation Between Personal Characteristics and Agility

This research question involves comparing the respondents agility score with that of several personal characteristics. The question states the following:

Is there a correlation between IT professionals' agility and their characteristics, i.e. their age, job function, experience and education?

First the agility score is compared with the age of the respondents to see if there is a correlation between the two. Figure 5.11 shows a scatter plot where the X-axis is the respondents' year of birth and the Y-axis is their agility score. The agility score for each year is calculated as the average of the scores for respondents born in that year. As the Figure illustrates the respondents' age ranges from 21 to 66 years but no apparent correlation exits in the data. The black line crossing the graph is a linear trendline and it clearly shows that there is no correlation between age and agility. This means that both young and old IT Professionals are equally agile and non-agile.

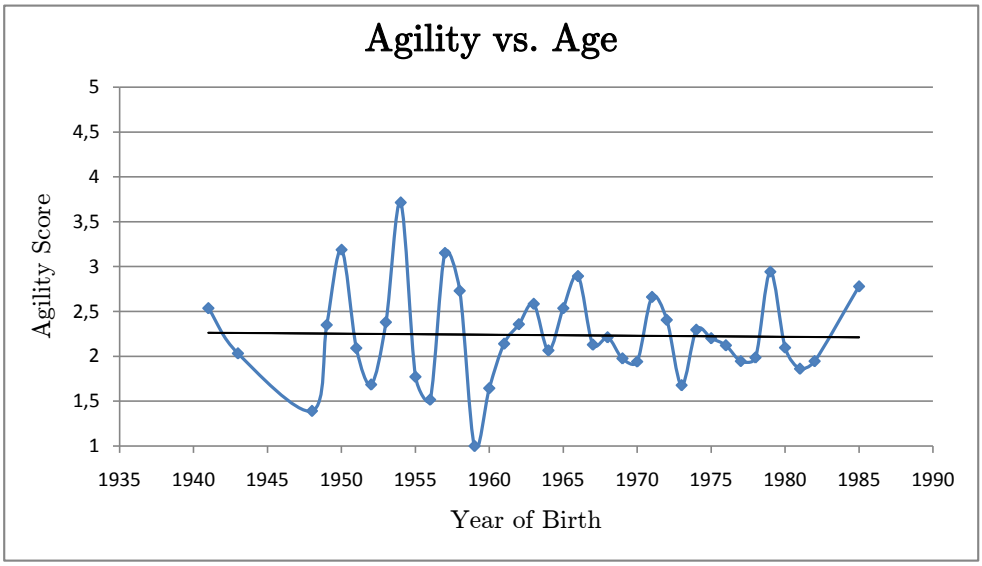


Figure 5.11: Scatter plot of respondents' agility compared with their age.

The second personal characteristic to compare with the agility score is experience. Figure 5.12 shows a scatter plot of the correlation between the two. As years of professional experience is closely related with the respondent's age it is not surprising that there also is no correlation between experience and agility. Again the trendline shows no indication that there exist a relationship with agility and experience.

Figure 5.12: Scatter plot of respondents' agility compared with their professional experience.

The third characteristic is education. Figure 5.13 illustrates the relationship between agility and education in a column chart. Each column color represents a different education, which are divided into the three agility intervals. The figure indicates that there exist a difference in the level of agility between respondents with longer educations and those with short course based education. However, as the number on the columns show, the course based education category is based on very few respondents, so this is likely to affect the results.

This means that our data support no correlation between education and agility. Also the questionnaire included several other education categories, like self-taught and Ph.D., but these were excluded to simplify the column graph and because they did not reveal anything significant.

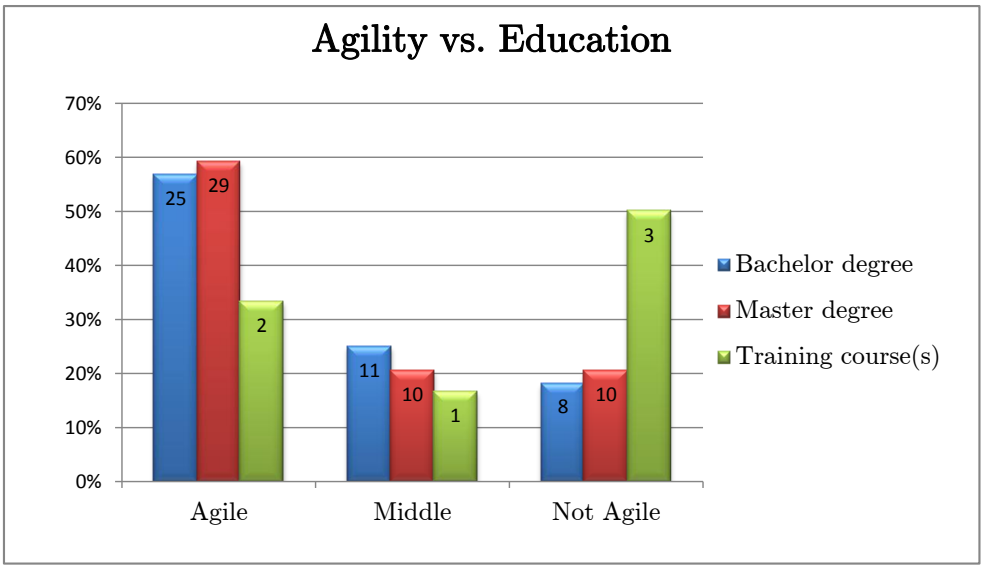


Figure 5.13: Column graph of respondents' agility compared with their education.

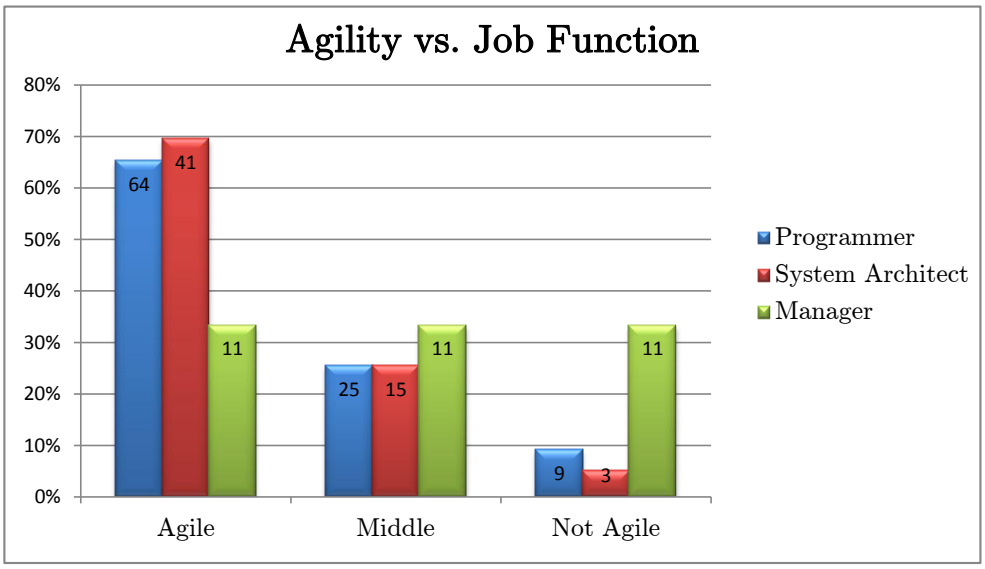


Figure 5.14: Column graph of respondents' agility compared with their job function.

The final personal characteristic is job function. Figure 5.14 shows three of the six job functions included in the questionnaire; Programmer, System Architect and Manager. These were selected because they are based on the majority of the respondents and reveal the most interesting results. As the figure shows there exist a great deal of difference between programmers, system architects and managers. Both programmers and system architects are highly agile while managers are distributed equally on the agility scale. The reason for this difference may be that managers have closer contact with the customers, who make

it difficult to adhere to the agile concepts. It may also be caused by managers having to consider planing and administrative issues that oppose the agile values.

The data from the main survey indicate that there is a correlation between a respondent's job function and their level of agility. 65% of the programmers and 69% of system architects are agile, while only 33% managers are agile.

5.4.3 IT Professionals, Their Team, Organization and Customer

This research question takes the other three perspectives of the bipolar preferred questions and compares them with the agility of IT professionals. The question states the following:

Is there a difference in the degree of agility of IT professionals and the agility of their team, organization and customer?

To find out if the agility of the other perspectives, team, organization or customer is different from that of IT professionals, they are analyzed in the same way as IT professionals are in the first research question. This means that the agility score is calculated based on the values of each perspective and the variance is used to measure how conflicted these answers are.

Figure 5.15 illustrates the agility of the team, which is very similar to that of IT professionals in Figure 5.10. When comparing the two it is clear that both perspectives have the same level of agility, where the majority of respondents reside on the agile side of the column graph.

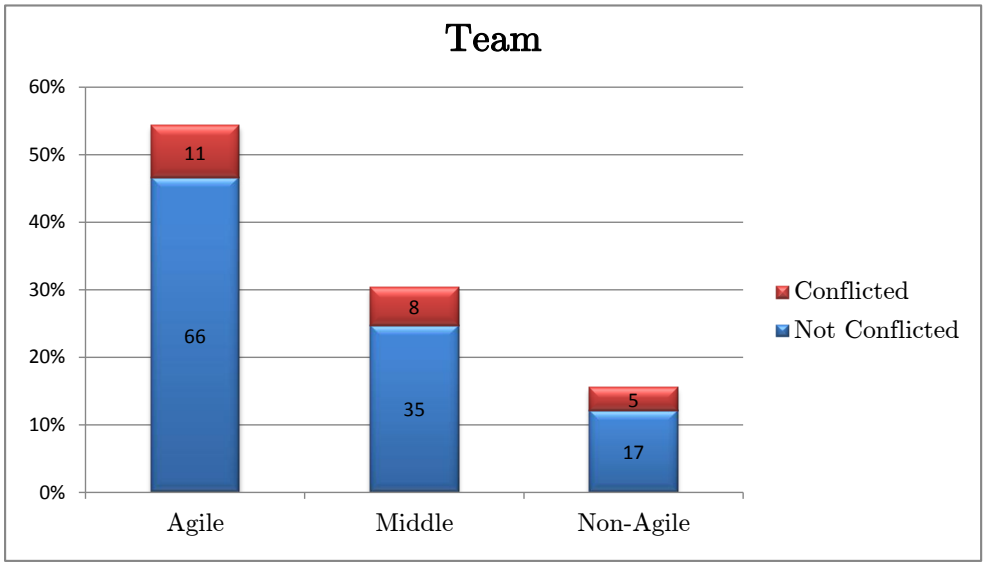


Figure 5.15: Column graph of the agility of Teams.

Next the organization is compared with IT professionals. Figure 5.16 shows the agility of the organization. Compared with Figure 5.10 it is clear that the majority are no longer on the agile side but is located both in the middle and on the non-agile side of the chart. This indicates that organizations are less agile than the IT professionals who work for them.

This being said, it is important to note that the values retrieved from the three other perspectives are estimations by the respondents themselves. This means that biases toward the organization and subjective answers can result in erroneous data, which in turn skews the conclusions drawn from them. However, laking the possibility of collecting completely objective data on each of the respondent's team, organization and customer, the method of allowing the respondent to estimate the three other perspectives is satisfactory. The strong differences in Figure 5.16 compared with Figure 5.10 indicate that there is a difference in the degree of agility between organizations and IT professionals.

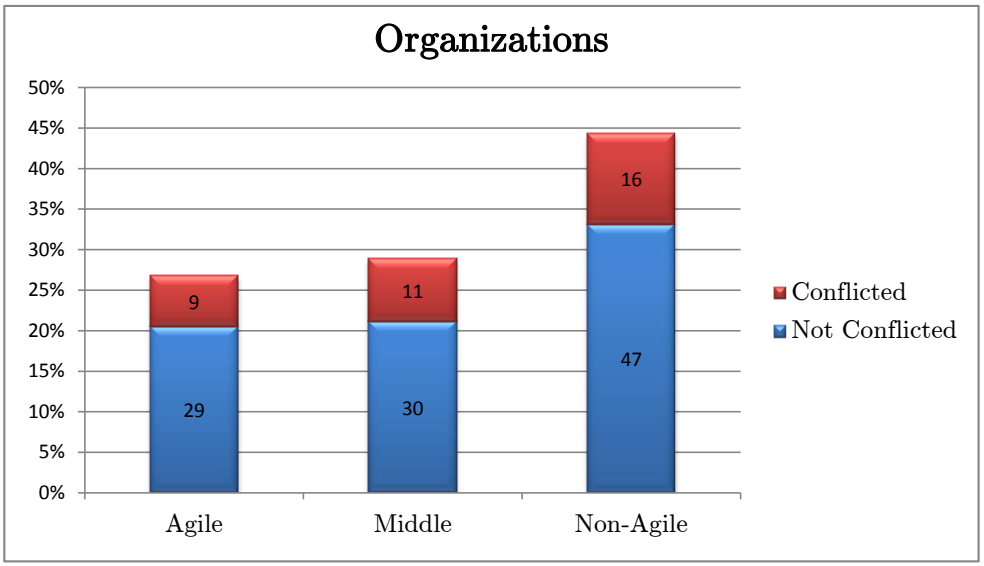


Figure 5.16: Column graph of the agility of Organizations.

The last perspective is the customer. The customer is an important part of Agile Software Development and it is therefore very interesting to see if the customer prefers the agile concepts over the non-agile concepts. If for example the customer requires a comprehensive up-front contract that documents and specifies every aspect of the system, it is very difficult for the IT professionals, the team or the organization to use the agile approach. Also agility requires a great deal of customer contact, which in turn requires that the customer is willing to get involved in the project.

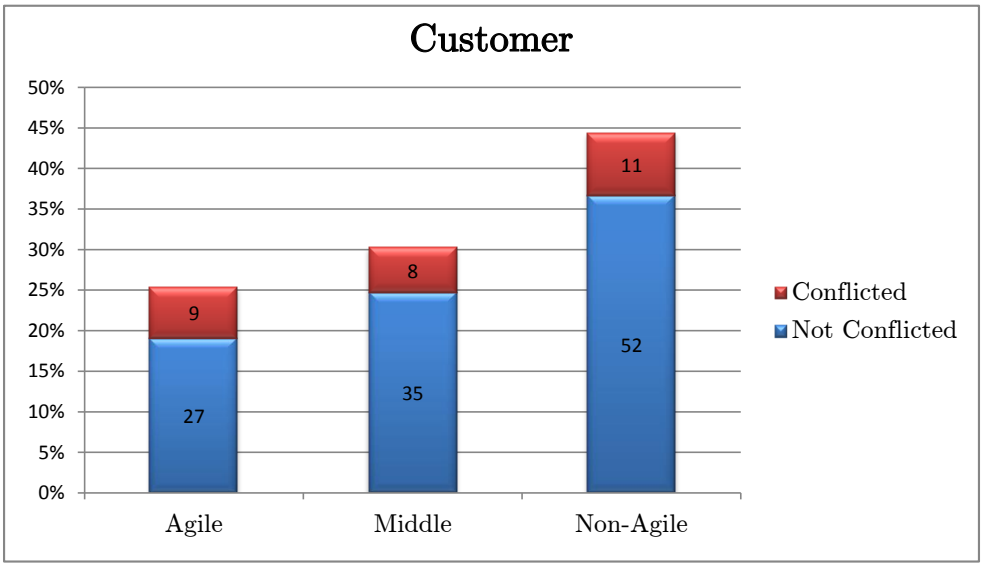


Figure 5.17: Column graph of the agility of Customers.

Figure 5.17 shows much the same trends as the organization, where the majority is located on the non-agile side of the agility scale. This means that there exists an imbalance between the agility of IT professionals and the customers, which in turn can lead to problems for the reason stated above.

5.4.4 Context of Software Projects

The final research question seeks to uncover if the context of software projects in general facilitate the use of Agile Software Development processes. This is important to know because it indicates if it is indeed possible for the four perspective groups to be agile in practice. If the context does not support the use of agile values and principles it will inevitably manifest itself in the results drawn from the bipolar preferred questions. The questions states the following:

Is the context of software projects suited for Agile Software Development?

The context factors used to determine project suitability for Agile Software Development are the five factors developed by Boehm and Turner [15] discussed in Chapter 2. These five factors consist of; personnel, dynamism, culture, size and criticality. First the five factors are presented individually and then concluded upon in a polar chart.

Boehm and Turner's personnel factor relies on a modified version of a categorization developed by Cockburn, which divides people into various skill levels. This categorization system was difficult to communicate successfully through at questionnaire and instead substituted with a question asking how many years of experience a respondent has. This question was then translated into two categories, experienced and non-experienced personnel by the use of a cutoff line. This delimiter was set to 5 years and resulted in 86% falling within the experienced category and 14% within the non-experienced category.

Figure 5.18 shows the results for dynamism, which Boehm and Turner define to percentage

of requirement change per month. Here the top score is 10% requirement change per month and most of the respondents fall bellow the 30% mark.

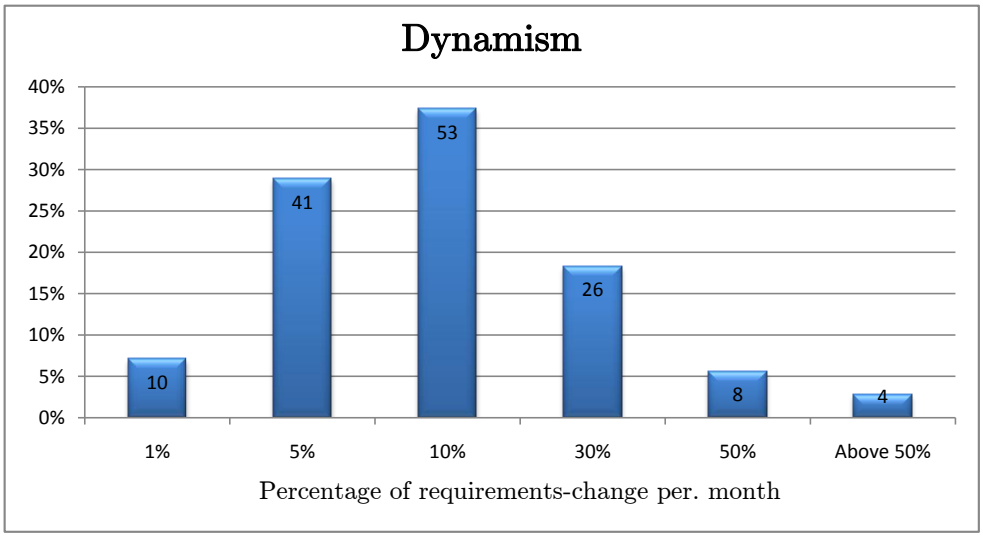


Figure 5.18: Column graph of the context factor Dynamism

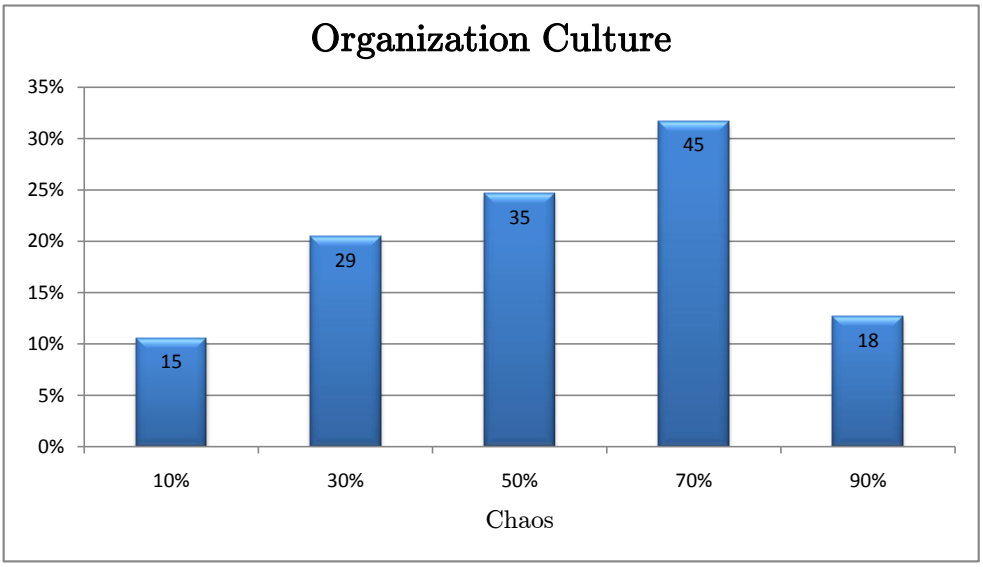


Figure 5.19: Column graph of the context factor Culture.

Figure 5.19 illustrates the culture factor. This factor measures if the culture at the respondents workplace thrives on chaos or order. The figure shows that the majority thrives on 70% chaos, which in turn means 30% order. The majority of respondents thrive on higher than 50% chaos.

Figure 5.20 shows the size of the development teams. These teams are all mostly small with less than 10 people. Non of the respondents in the survey work in teams of more than 100 people.

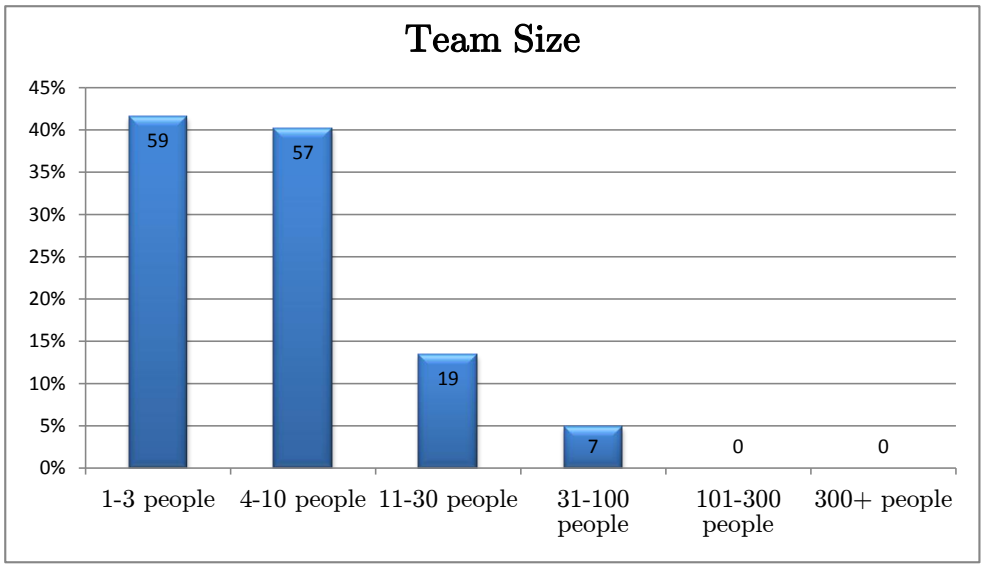


Figure 5.20: Column graph of the context factor Team Size.

Figure 5.21 show the last of Boehm and Turner's context factors, which is the criticality of the software. The five columns in the graph display what will happen if an error occurs in the software. The majority of respondents work on software that will cost the company discretionary money but not result in loss of life or essential money that can cause bankruptcy.

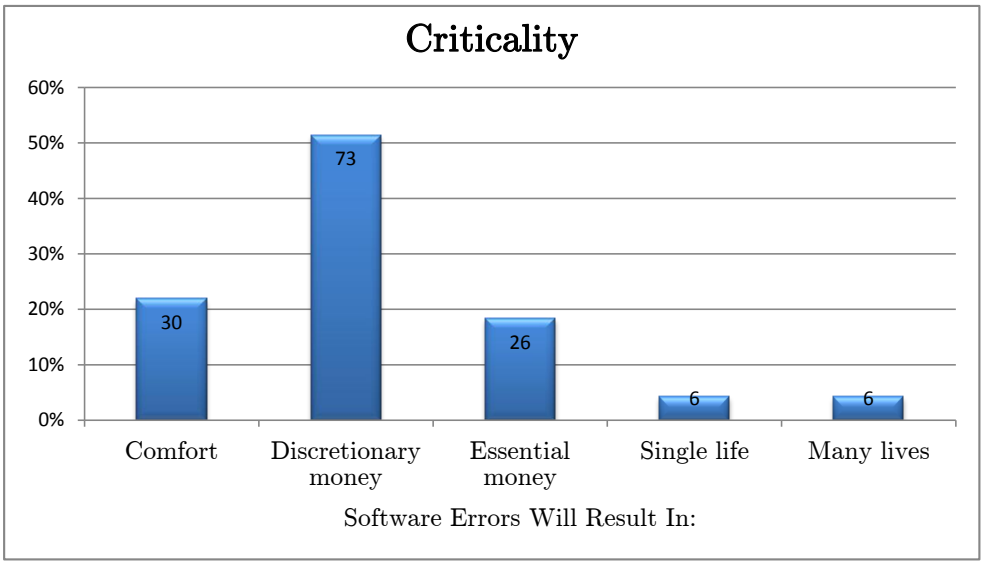


Figure 5.21: Column graph of the context factor criticality.

To conceptualize the five context factors Figure 5.22 shows a five axis polar chart. Each axis on the chart represents one of the five context factors. As illustrated in Figure 4.8 each context factor has its own individual measurement unit on a scale from 1 to 5. Due to technical reasons this could not be fitted to Figure 5.22 but each scale is present in the column graph corresponding to the context factor. So the culture axis has a scale of 10%, 30%, 50%, 70% and 90%, and the same goes for the other four axes.

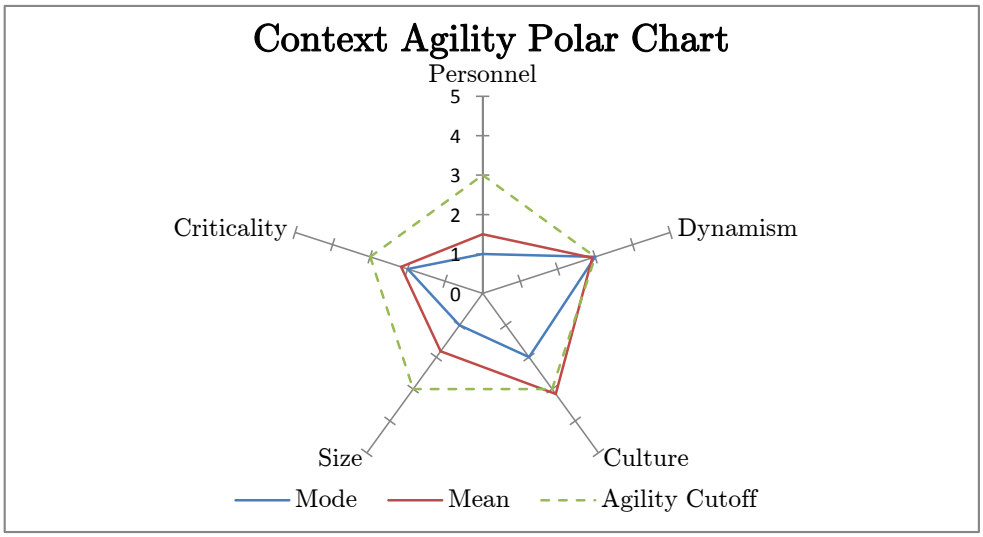


Figure 5.22: Polar chart of the five context factors from Boehm and Turner[15].

Figure 5.22 illustrates three separate elements. The green line represents the cutoff between agile and plan-driven home-grounds as explained by Boehm and Turner, where agility is within the center and plan-driven is towards the edge of the chart. The line is dotted because there exist no clear cut definition of when it is most appropriate to use agile or plan-driven methods. It is up to the organization to decide which approach best suites the needs they have, but the line can be used as a reference point.

The blue line represents the mode of the respondents' answers. This indicates that the answer most frequently given by respondents to a context factor is plotted in the chart. This shows that the context factor of most respondents is clearly within the realm of the agile home-ground. The only point where the mode line approaches the plan-driven home-ground is on dynamism because most respondents have very few requirements changes.

The red line in Figure 5.22 is the mean of the respondents' answers to the five context question. This takes into account not only the most frequently used answer but takes an average of them all. This makes the context considerably less suited to agile development but it is still within the limits of the agile home-ground cutoff line.

As a conclusion to this research question the project contexts of the vast majority of respondents fall within the agile home-ground and is therefor well suited for Agile Software Development.

6 Discussion

This chapter presents a discussion of our research project. It explains possibilities and complication with future research and describes the lessons learned.

6.1 Operationalizing Agility

The examination of the Agile Manifesto and review of the four agile authors were difficult to manage and it was difficult to present the results in a clear and understandable manner. Describing a single principle, would often result in long complex explanations that overlapped the other principles and values. This meant that presenting the results in a structured way was difficult.

It is believed that measuring the agility of IT professionals can not be done without employing the Agile Manifesto and its values and principles in some way. However, it can be questioned whether the seven areas identified in Chapter 3 and used in the questionnaire are the most suitable for measuring agility. Some of the areas might be to blurry and require more refinement. The *Iterative development* versus *Plan-driven* question is a good example on this. In the questionnaire the description of *Iterative development* does not include *frequent deliveries* or *early deliveries*, which both are emphasized as important by agile authors. Here we simply made a choice to limit the description of *Iterative development* to a more readable size. A better alternative would perhaps be to split the question and ask the respondent to choose between different properties, e.g *frequent deliveries of software* versus *stable requirements from the customer*. The underlying assumption would be that an agile minded person would prefer the first. He knows that deliveries of software are just as important internally (to the development team) as to the customer. Internal deliveries of software allow everyone on the team to watch and learn from a growing product. Stable requirements on the other hand will not.

6.2 Questionnaire Design

Our skills in questionnaire design and administration were at a novice level when beginning the project and has improved significantly during the project. However, when looking back, several things could have been handled more appropriately. Small mistakes include ambiguous questions which was not caught during the field test of the questionnaire. A more crucial error was that the bipolar preferred questions did not provide the respondent with a "Don't know" or "The concepts can not be compared" option. As respondents were forced to state their answer on the 1-5 likert scale or skip the question, this could have introduced clear bias. E.g. respondents who felt that "Working Software" and "Documentation" can not be valued over each other, would properly skip the question or select the middle option.

If a similar approach for measuring agility should be carried out or if work on the instrument should be continued, the use of judges to weigh the agility of the seven areas would be beneficial. A benchmark for agility was established by conducting the survey among an agile user group. However, a more precise measurement tool could be developed if the different measurement areas of agility were also weighted and given a score. This could be done by invoking a discussion group of agile experts and ask them to arrive at some ranking of and/or score for each area.

As mentioned in the Section 4.1 the method we used to gather respondents are somewhat problematic. The problem lies in, that we do not have any control of who participates and who many persons that receives our questionnaire. Thus we cannot precisely calculate the response rate. As response rate is an indicator for reliability, this should be addressed if the process should be done again. To address the problem of who participates, we included a job function question in the questionnaire. This helped filter and use only those that are within the target population and remove all others. However, this still do not solve the problem with regard to how many that of the target population got the questionnaire. Different approaches could be used to avoid this problem, e.g. by selecting a randomized sample within the target population, thus knowing exactly how many gets the questionnaire and how many have participated, hence it is possible to calculate the response rate more accurately.

When we started to conduct our main survey, we observed that a lot of respondents did not complete the questionnaire, i.e. they did not answer all the questions. We tried to identify why, and it was noticed that most of the respondents did quit the questionnaire when they reached the bipolar preferred questions. In the beginning we had all seven questions on one page. This could have been very overwhelming for the respondents as, each question contain a lot of text and information and answer options. Here, the power of using an online questionnaire using SurveyXact survey tool, it was possible to change this, so that each page only contained 2 questions. This actually improved the number of completed questionnaires, however, this was noticed to late in the process. The thing we learn from this experience was that it is very important to have a very simple and easy design, that do not scare away the respondent. This will indeed affect the participation.

7 Conclusion

This report argues that Agile Software Development is an interesting and relevant subject within Information Systems – Software Development. Based on a review of nine articles and three survey reports it is concluded that agile practices and methods are used throughout the industry and that empirical evidence supporting the value of the agile methodology exists. However, it is also identified that a clear definition of Agile Software Development and what it entails to be agile is missing. The review established that the survey reports focuses on the use of agile practices and methods but lacks validation of respondents' use and knowledge of the investigated practices and methods. Our study argues that agility, i.e. being agile, is difficult to measure through adherence to practices or methods, and that it is unreliable to do so. This report suggests that measuring respondents' attitude towards central agile values and principles using attitudinal scales from phycology provides a better alternative. This gave inspiration to four research questions:

- *How agile are IT professionals?*
- *Is there a difference in the degree of agility of IT professionals and the agility of their team, organization and customer?*
- *Is there a correlation between IT professionals' agility and their characteristics, i.e. their age, job function, experience and education?*
- *Is software projects' context suited for Agile Software Development?*

In order to answer the research questions, a research process model proposed by Kumar [6] inspired the development of an iterative research model consisting of eight steps. As a consequence of the model's iterative nature the specific content of the report and the artifacts produced has evolved through the entire process, and only the results are presented in the report.

In the third step of the research model, an instrument to measure agility and a questionnaire was developed. The instruments primary purpose is to measure the agility of IT professionals and its secondary purpose is to measure the agility of three other perspectives; *team*,

organization and *customer*. The instrument was constructed through a literature review of four notable agile authors, where the 12 agile principles were used as a framework. To complete the operationalization, seven areas where agility can be measured were identified. The seven measurement areas were used to construct the final instrument for measuring agility which is designed for a quantitative research study where a questionnaire is used. In order to ensure the validity of the research and answering the research questions the questionnaire design was done using questionnaire design techniques described by Kumar [6] and Trochim [41]. To answer the fourth question, four of Boehm and Turner's *five critical agility and plan-driven factors* [15] were transformed into questions in the questionnaire.

In step four, contact with external organizations was established, and an agreement for the distribution of the questionnaire was set up.

In step five, respondents from four organizations in total participated in the survey and data from 304 respondents was gathered of which a total of 142 could be used for analysis after item non-response respondents was removed.

Step six consisted of a benchmark of the instrument's primary purpose of measuring the agility of IT professionals. It was carried out using the Danish Agile User Group as a sample and indicated that six out of seven variables in the instrument performed satisfactory.

In step seven, the results were analyzed using factor analysis and well known statistical concepts such as mean, median, dispersion and mode. It is concluded that a large part (48%) of IT professionals are agile, 16% are of medium agility and 11% are not agile. The only characteristic of IT professionals which correlates with agility is job function. The analysis concludes that most programmers (65%) and system architects (69%) are agile, while only few managers (33%) are equally agile. The results also concludes that most teams are agile (46%), while a lower number of organizations (20%) and customers (19%) are agile and thereby a difference in agility between IT professionals and their organization and customers is established. Finally, empirical evidence supporting the fact that most projects are suited for Agile Software Development is presented.

In step eight, the project was discussed and communicated. It is addressed how the operationalization of Agile Software Development and the instrument for measuring agility could be improved and lessons learned from the design and execution of the questionnaire is elaborated on. In addition work done in the project concerning the relationship between the agile values, principles, methods and practices is communicated in an article for PROSIT.

List of Figures

List of Acronyms

AAU Aalborg University

ACM Association for Computing Machinery

DAUG Danish Agile User Group

DoD Department of Defence

ERS Emergency Response System

FSD Federal Software Devision

HCI Human-Computer Interaction

IEEE Institute of Electrical and Electronics Engineers

IID Iterative and Incremental Development

IT Information Technology

LOC Lines Of Code

MTBF Mean Time Between Failure

PC Principle Component

XP eXtreme Programming

XP-EF XP-Evaluation Framework

WOS Web of Science

References

References from Books

[1] Afifi, Clark, and May. *Computer-Aided Multivariate Analysis - Fourth Edition*. Chapman and Hall/CRC, 2004.

[2] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison Wesley Professional, 2004.

[3] Alistair Cockburn. *Agile Software Development*. Cockburn Highsmith Series Editors, 2000.

[4] James A. Highsmith. *Agile Software Development Ecosystems*. Pearson Education Inc, 2002.

[5] Jeffrey Rubin. *Handbook of Usability Testing: How to plan, design and conduct effective tests*. 1994.

[6] Ranjit Kumar. *Research Methodology - A Step-by-Step Guide for Beginners*. SAGE Publications Ltd, 2005.

[7] Sabine Landau and Brian S. Everitt. *A Handbook of Statictical Analyses using SPSS*. Chapman and Hall/CRC, 2004.

[8] Craig Larman. *Agile & Iterative Development - A Manager's Guide*. Addison-Wesley - Pearson Education Inc., 2004.

[9] Geoffrey Marczyk, David DeMatteo, and David Festinger. *Essentials of Research Design and Methodology*. John Wiley & Sons Inc. Hoboken New Jersey., 2005.

[10] Paul Newbold, William L. Carlson, and Betty Thorne. *Statistics for Business and Economics - Sixth Edition*. Pearson Prentice Hall, 2006.

[11] Mary Poppendieck and Tom Poppendieck. *Implementing Lean Software Development From Concept to Cash, Chapter 2*. Addison Wesley Professional, 2006.

[12] Roger S. Pressman. *Software Engineering - A Practitioner's Approach 6th Edition*. McGraw-Hill, 2006.

[13] Ledyard R. Tucker and Robert C. MacCallum. *Exploratory Factor Analysis*. http://www.unc.edu/~rcm/book/factornew.htm, 1997.

References from Articles

[14] Scott W Ambler. Survey says: Agile works in practice. *Dr. Dobb's Journal*, 2006.

[15] Barry Boehm and Richard Turner. Using risk to balance agile and plan-driven methods. *IEEE Computer Society*, 2003.

[16] Tom DeMarco and Barry Boehm. The agile methods fray. *IEEE Computer Society*, 2006.

[17] Martin Fowler. Is design dead? *Dr. Dobb's Journal*, 2001.

[18] Ann Fruhling and Gert-Jan De Vreede. Field experiences with extreme programming: Developing an emergency response system. *Journal of ManagementInformation Systems*, 2006.

[19] R.L. Glass. Agile versus traditional: make love not war. *Cutter IT Journal 14 (12) 1218.*, 2001.

[20] Peter Gould. What is agility? *Manufacturing Engineer*, 1997.

[21] Vamsidhar Guntamukkala, H. Joseph Wen, and J. Michael Tarn. An empirical study of selecting software development life cycle models. *IOS Press*, 2006.

[22] Christina Hansson, Yvonne Dittrich, Bjorn Gustafsson, and Stefan Zarnak. How agile are industrial software development practices? *The Journal of Systems and Software 79 Elsevier*, 2006.

[23] Lucas Layman, Laurie Williams, and Lynn Cunningham. Motivations and measurements in an agile case study. *Journal of Systems Architecture 52 Elsevier*, 2006.

[24] Lucas Layman, Laurie Williams, and William Krebs. Extreme programming evaluation framework for object-oriented languages version 1.3. *North Carolina State University Department of Computer Science Raleigh NC TR-2004-11*, 2004.

[25] Mikael Lindvall, Vic Basili, Barry Boehm, Patricia Costa, Kathleen Dangle, Forrest Shull, Roseanne Tesoriero, Laurie Williams, and Marvin Zelkowitz1. Empirical findings in agile methods. *Springer-Verlag Berlin Heidelberg*, 2002.

[26] Donald J. Reifer. How good are agile methods? *IEEE Software*, 2002.

[27] Jeremy Rose. Soft systems methodology as a social science research tool. *Manchester Metropolitan University*, 2005.

[28] Laurie Williams and Alistair Cockburn. Agile software development: Its about feedback and change. *Computer Volume 36 Issue 6 June 2003*, 2003.

References from Homepages

[29] Alistair Cockburn. IT Conversations - Agile Software Development. http://www.itconversations.com/transcripts/175/transcript-print175-1.html, 2004.

[30] Association for Computing Machinery. Association for Computing Machinery. , 2007.

[31] Digital Focus. Agile 2006 survey. http://www.digitalfocus.com/, 2006.

[32] Institute of Electrical and Electronics Engineers. Institute of Electrical and Electronics Engineers. http://www.ieee.org/web/aboutus/today/index.html#xplore, 2007.

[33] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, ... The Agile Manifesto. http://www.agilemanifesto.org/, 2001.

[34] Martin Fowler. The New Methodology. http://www.martinfowler.com/articles/newMethodology.html, 2005.

[35] Merriam Webster. Merriam Webster. http://www.merriam-webster.com/dictionary, 2007.

[36] Version One. The state of agile development. http://www.versionone.net/surveyresults.asp, 2006.

[37] Standish Group. The CHAOS Report (1998). http://www.velocitystorm.com/resources/chaos.pdf, 1998.

[38] Shine Technologies. Agile methodologies. http://www.agilealliance.org/system/article/file/1121/file.pdf, 2003.

[39] Web of Science. Web of Science. http://scientific.thomson.com/products/wos, 2007.

[40] WikiPedia. Normal Distribution Graph. http://upload.wikimedia.org/wikipedia/commons/1/1b/Normal_distribution_pdf.png, 2007.

[41] William M.K. Trochim. Web Center for Social Research Methods. http://www.socialresearchmethods.net/, 2007.

All URLs are valid as of June 8^{th}, 2007.

Resume

Vores projekt omhandler Agil Softwareudvikling og hvad det vil sige at være agil. Der foretages en kvantitativ undersøgelse af IT professionelle (udviklere, projektledere, systemarkitekter, osv.) for at undersøge deres agilitet gennem deres holdning. Dette gøres ud fra en detaljeret undersøgelse af værdierne og principperne bag Agil Softwareudvikling. Målet er at udvikle et måle instrument, der gennem et spørgeskema er i stand til at svare på vores forsknings spørgsmål, der lyder som følger:

- Hvor agil er IT professionelle?
- Er der en forskel i graden af agilitet hos IT professionelle og agiliteten af deres udviklingshold, organisation og kunde?
- Er der en correlation mellem IT professionelles agilitet og deres person karakteristika, f.eks. deres alder, job funktion, erfaring og uddannelse?
- Er software projekters kontekst egnet til Agil Softwareudvikling?

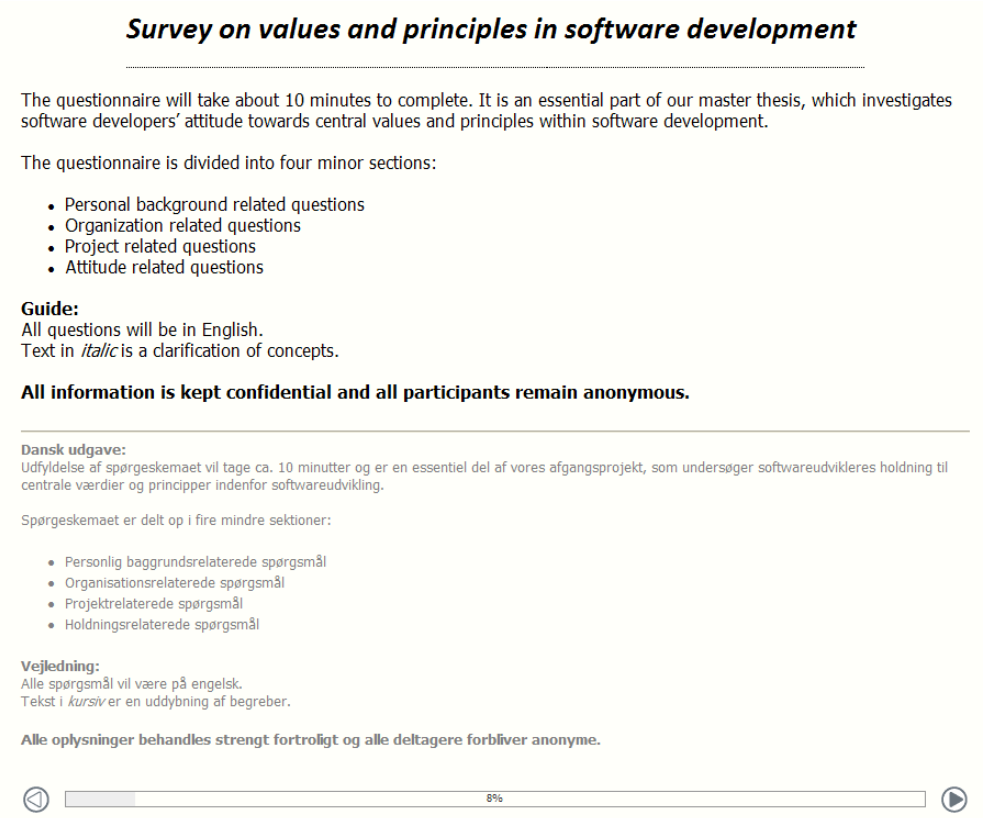
Rapporten opstiller først en processmodel for projektet, hvor der lægges vægt på dens iterativitet. Herefter foretages undersøgelse af relateret litteratur i form af artikler og andre spørgeskemaundersøgelser. Her konstateres det, at agile metoder og praktikker ikke egner sig til at måle agilitet gennem et spørgeskema. Det fastlås også at størsteparten af det undersøgte litteratur og specielt spørgeskema undersøgelserne kun undersøger brugen af enkelte metoder og praktikker eller indeholder en overfladisk gennemgang af det agile manifest og de 12 agile principper. Det afgøres derfor at undersøge agilitet ud fra respondenternes holdning til agile værdier og principper, fremfor et forsøg på at afdække deres egentlige anvendelse af agile metoder og praktikker.

Ud fra en undersøgelse af de agile vædier identificerer vi syv områder, hvor agilitet kan måles og som udgøre de centrale aspekter i agile softwareudvikling. De syv områder bliver omdannet til et instrument til måling af agilitet gennem holdning, ved at benytte likert skaler. Det udviklede instrument bliver indkluderet i et spørgeskema, som sendes ud til respondenterne via e-mail. Spørgeskemaet er således et online spørgeskema, hvilket gør indsamling af data betydeligt nemmere. Det anvendte værktøj er SurveyXact.

Dette munder ud i en spørgeskemaundersøgelse blandt IT professionelle i Danmark fordelt på 85 virksomheder og 142 respondenter. Det indsamlede data bliver analyseret gennem faktor og komponent analyse og anvendelse af almindelige statistiske redskaber så som middelværdi, median, gennemsnit. Desuden foretages en benchmarktest ved hjalp af en agil brugergruppe. Testen bliver brugt til at finde en agilitets faktor, som anvendes i besvarelsen af forskningsspørgsmålene.

Det konkluderes i rapporten at størstedelen af IT professionelle (48%) er agile og at de fleste programmører og systemarkitekter også er agile. Ydermere konkluderes det også, at de fleste udviklingshold også er agile (46%). Derimod er blot 20% af organisationerne og 19% af deres kunder agile. Endeligt konkluderes det at gennemsnittet af projekternes kontekst er egnet til Agil Software Udvikling.

B Questionnaire



Survey on values and principles in software development

The questionnaire will take about 10 minutes to complete. It is an essential part of our master thesis, which investigates software developers' attitude towards central values and principles within software development.

The questionnaire is divided into four minor sections:

- Personal background related questions
- Organization related questions
- Project related questions
- Attitude related questions

Guide:
All questions will be in English.
Text in *italic* is a clarification of concepts.

All information is kept confidential and all participants remain anonymous.

Dansk udgave:
Udfyldelse af spørgeskemaet vil tage ca. 10 minutter og er en essentiel del af vores afgangsprojekt, som undersøger softwareudvikleres holdning til centrale værdier og principper indenfor softwareudvikling.

Spørgeskemaet er delt op i fire mindre sektioner:

- Personlig baggrundsrelaterede spørgsmål
- Organisationsrelaterede spørgsmål
- Projektrelaterede spørgsmål
- Holdningsrelaterede spørgsmål

Vejledning:
Alle spørgsmål vil være på engelsk.
Tekst i *kursiv* er en uddybning af begreber.

Alle oplysninger behandles strengt fortroligt og alle deltagere forbliver anonyme.

8%

Figure B.1: Page 1 of the questionnaire

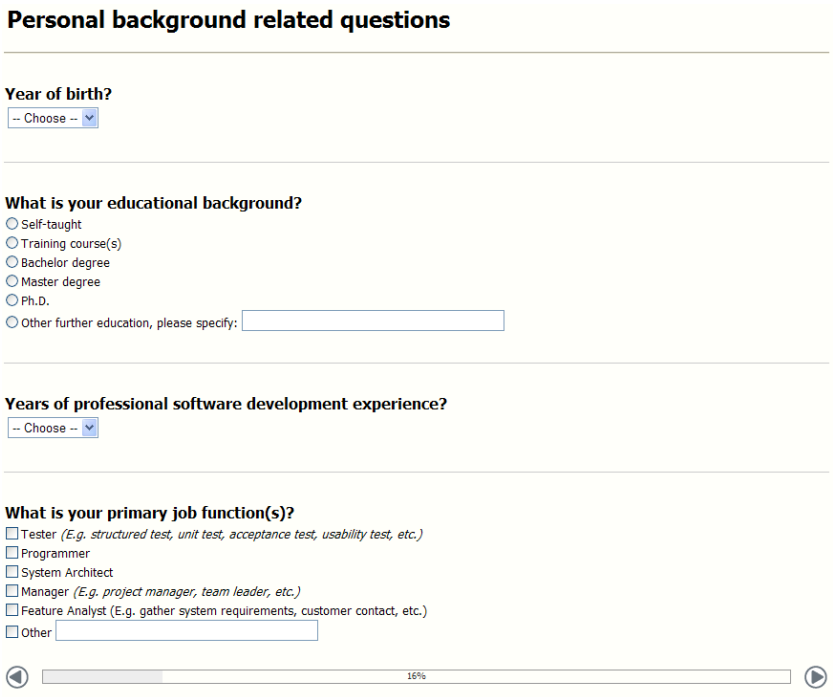


Figure B.2: Page 2 of the questionnaire

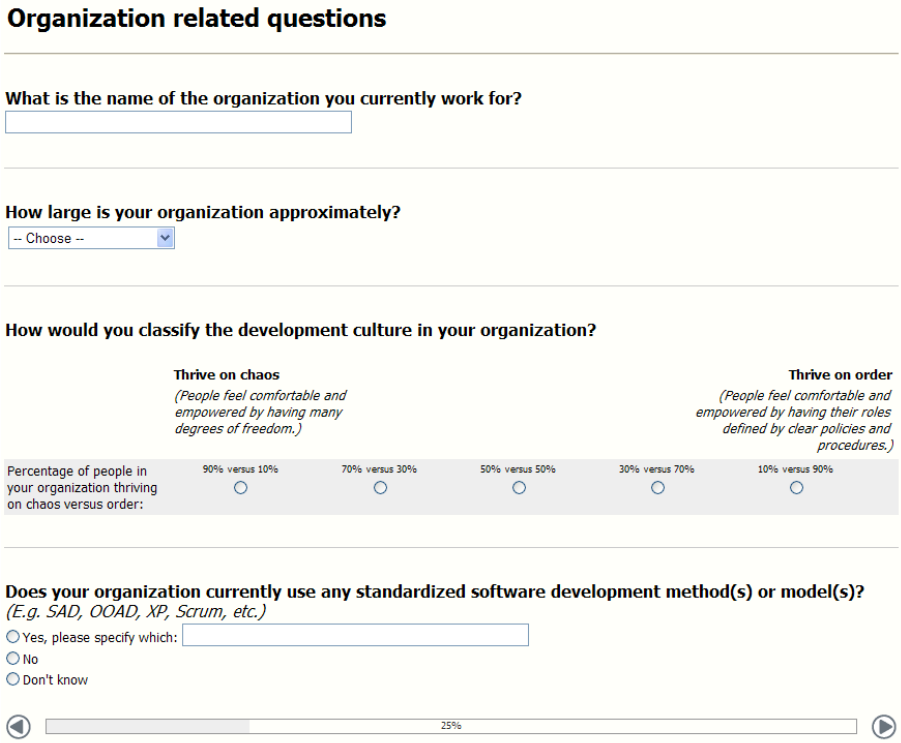


Figure B.3: Page 3 of the questionnaire

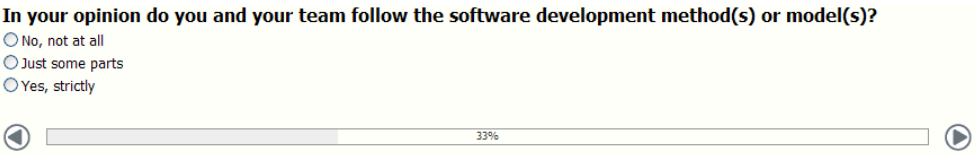


Figure B.4: Page 4 of the questionnaire

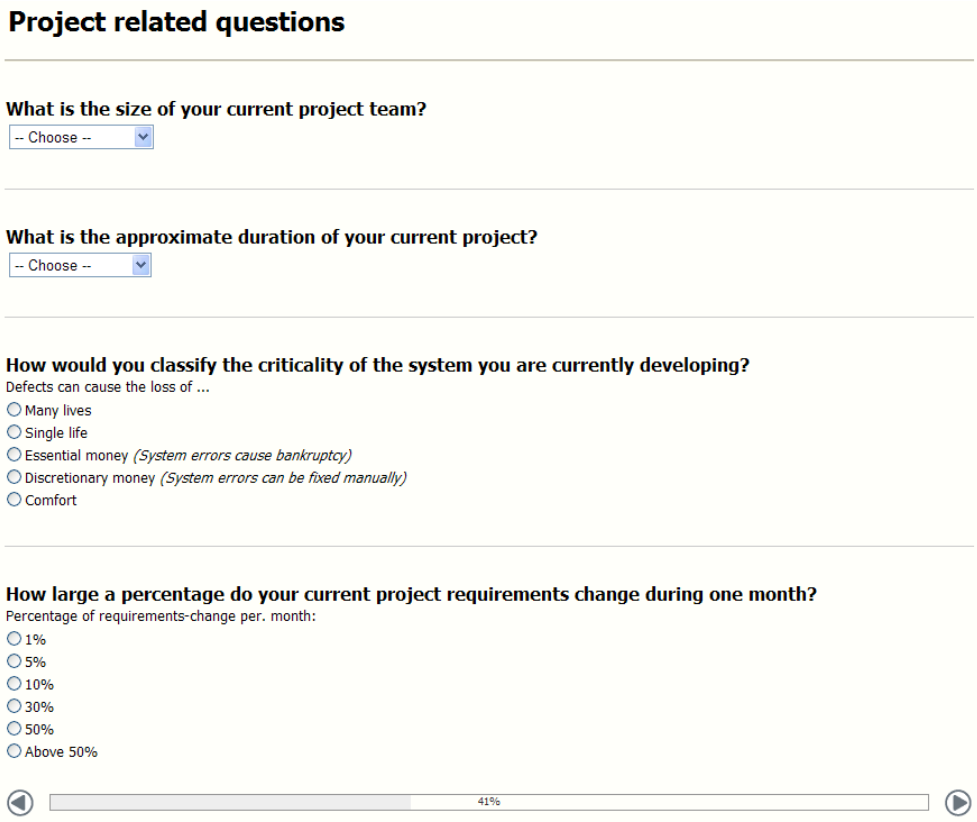
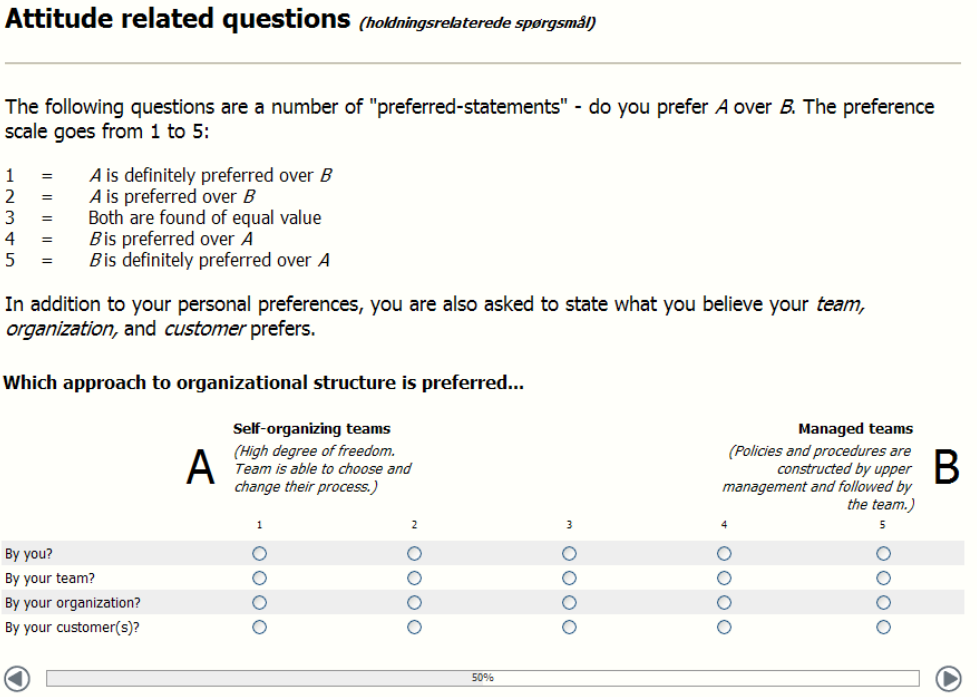


Figure B.5: Page 5 of the questionnaire



Attitude related questions *(holdningsrelaterede spørgsmål)*

The following questions are a number of "preferred-statements" - do you prefer *A* over *B*. The preference scale goes from 1 to 5:

1 = *A* is definitely preferred over *B*
2 = *A* is preferred over *B*
3 = Both are found of equal value
4 = *B* is preferred over *A*
5 = *B* is definitely preferred over *A*

In addition to your personal preferences, you are also asked to state what you believe your *team, organization,* and *customer* prefers.

Which approach to organizational structure is preferred...

	A	1	2	3	4	5	B
	Self-organizing teams *(High degree of freedom. Team is able to choose and change their process.)*						**Managed teams** *(Policies and procedures are constructed by upper management and followed by the team.)*
By you?		○	○	○	○	○	
By your team?		○	○	○	○	○	
By your organization?		○	○	○	○	○	
By your customer(s)?		○	○	○	○	○	

50%

Figure B.6: Page 6 of the questionnaire

Which approach to requirements handling is preferred...

	A	1	2	3	4	5	B
	Continuous customer collaboration *(People with business knowledge work close and continuously with the development team.)*						**A software requirements specification** *(Requirements are gathered, documented and distributed to developers through a software requirements specification.)*
By you?		○	○	○	○	○	
By your team?		○	○	○	○	○	
By your organization?		○	○	○	○	○	
By your customer(s)?		○	○	○	○	○	

When measuring project progress, which approach is preferred...

	A	1	2	3	4	5	B
	Working software *(A number of integrated and tested features.)*						**Documentation** *(Finished intermediate documents such as test reports, review documents, UML diagrams, etc.)*
By you?		○	○	○	○	○	
By your team?		○	○	○	○	○	
By your organization?		○	○	○	○	○	
By your customer(s)?		○	○	○	○	○	

58%

Figure B.7: Page 7 of the questionnaire

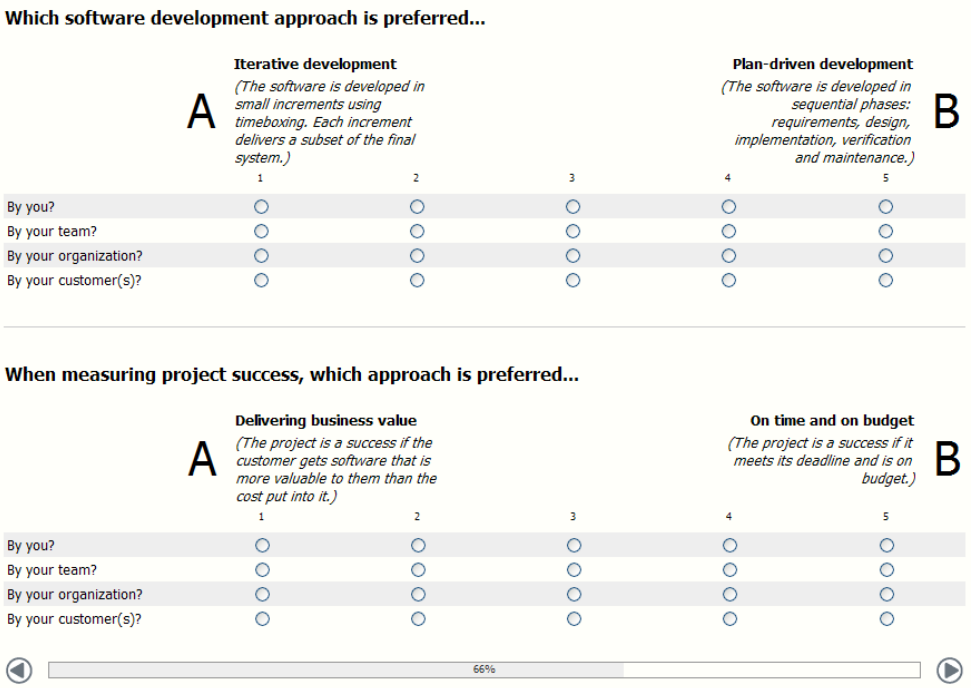


Figure B.8: Page 8 of the questionnaire

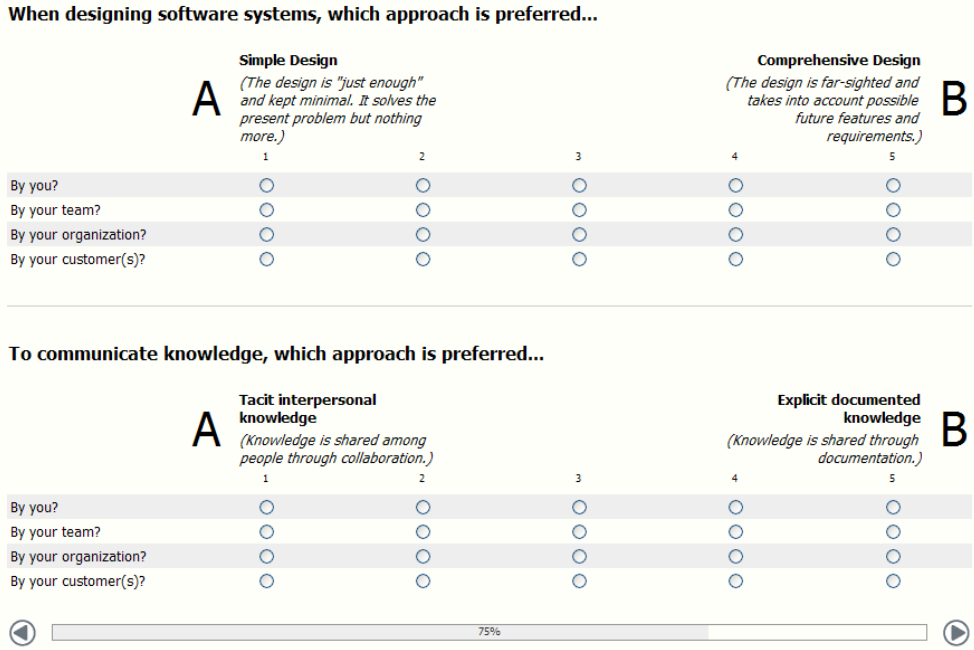


Figure B.9: Page 9 of the questionnaire

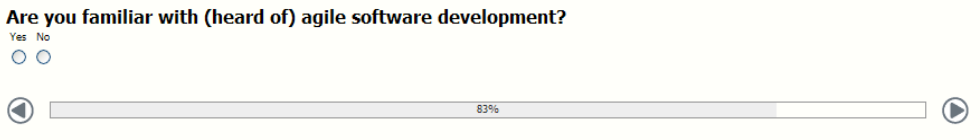


Figure B.10: Page 10 of the questionnaire

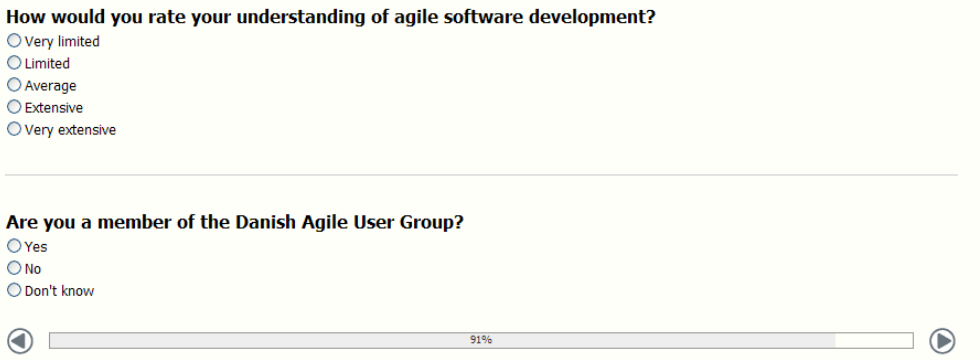


Figure B.11: Page 11 of the questionnaire

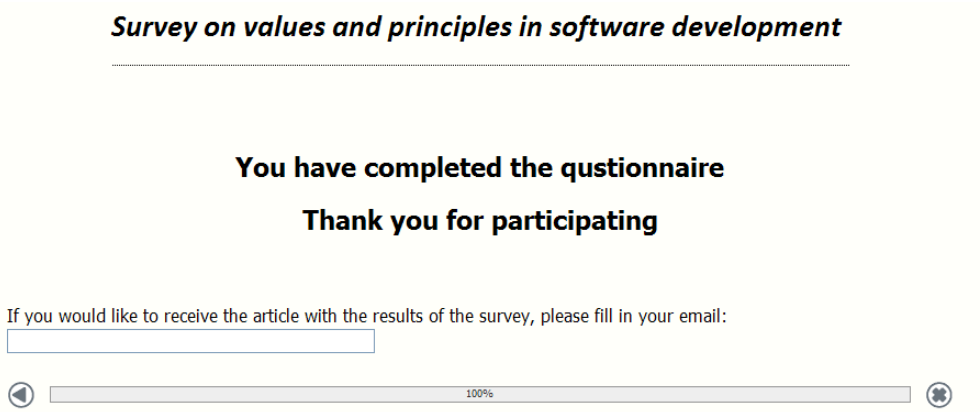


Figure B.12: Page 12 of the questionnaire

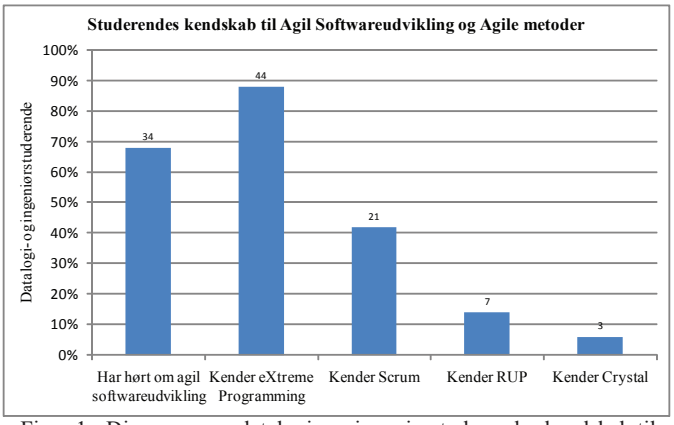
C Agil Forvirring

Artikel skrevet til magasinet ProsIT's Juni nummer.

Der hersker ofte forvirring når emnet er Agil Softwareudvikling. Hvad gemmer der sig egentligt bag dette buzz-ord? Er det mere end bare en samling metoder med eXtremme programming som flagskib? Denne artikel er en smutvej fra forvirring til forståelse.

C.1 Studerendes Kendskab til Agil Softwareudvikling

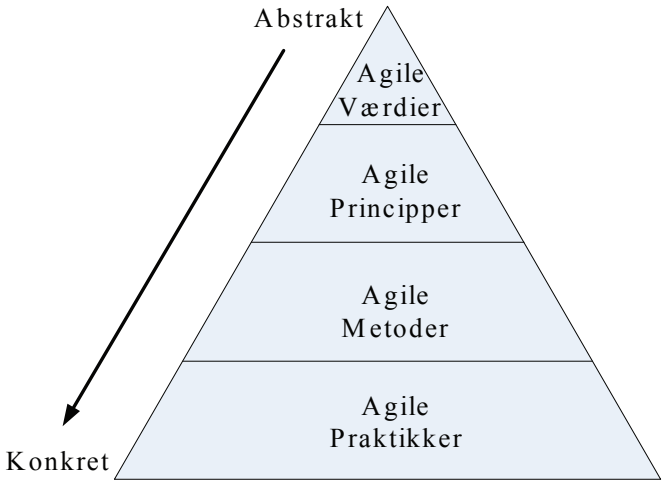
I en undersøgelse af datalogi- og ingeniørstuderendes kendskab til Agil Softwareudvikling, som vi foretog på Aalborg Universitet, viste det sig bl.a., at 68% havde hørt om agil softwareudvikling (se Figur C.1). Mere interessant var det dog, at 88% af de studerende kendte til den agile udviklingsmetode eXtreme Programming (XP). Det betød, at der faktisk var 20% af de studerende, som ikke havde hørt om Agil Softwareudvikling, men godt nok kendte den agile udviklingsmetode. At XP er så populær, er måske ikke overraskende. I blot 12 enkle praktikker beskriver den, hvordan der indsamles krav, designes, konstrueres og testes i et software projekt. Dette gør den utrolig tilgængelig og interessant. Men populariteten har dog også den sideeffekt, at mange er hurtige til at forbinde praktikker i XP med Agil Softwareudvikling og sætter lighedstegn mellem de to. Agil Softwareudvikling er dog meget mere end bare praktikker og metoder.



Figur 1 - Diagram over datalogi- og ingeniørstuderendes kendskab til Agil Softwareudvikling og metoder. I alt deltog 50 respondenter.

C.2 Agil Softwareudvikling

Udtrykket "Agil Softwareudvikling" blev dannet i 2001, da 17 software- og metodeudviklere mødtes på et skisportssted i Snowbird Utah. Til mødet deltog metodeforfattere af bl.a. XP, Scrum og Crystal, som alle i dag går under betegnelsen agile metoder. Førhen var metoderne kendt under betegnelsen "letvægtsmetoder", og blev udviklet som et led i et opgør med de traditionelle plan-drevne metoders' konstante fokus på dokumentering og infleksible proces. Til mødet i Snowbird blev ideerne bag disse metoder forenet under et fælles set af værdier, som er udtrykt i det Agile Manifest. Manifestet fremsætter fire kærneværdier, som udviklingsmetoder skal opnå for at kunne betragtes som agile. Udover manifestet fremsatte man også 12 agile principper, som uddyber værdierne. Agil softwareudvikling består altså ikke kun af udviklingsmetoder og praktikker, men også af værdier og principper. Dette er illustreret på Figur C.2. Figuren viser at Agil Softwareudvikling, som koncept kan opdeles i 4 dele; værdier, principper, metoder og praktikker. Værdier er de vigtigste, men også mest abstrakte og dermed placeret i toppen af pyramiden. Jo længere man bevæger sig ned, jo mere konkret bliver det.



Figur 2 - Model over Agile Softwareudvikling

C.2.1 Agile Værdier

Manifestet er vist på Figur C.2.1 og udtrykker de 4 værdier, som forfatterne blev enige om var vigtige for dem og dermed i Agil Softwareudvikling. Værdierne er dog så abstrakte, at de er åbne for fortolkning. For eksempel kan værdi nr. 2 ("Working software over comprehensive documentation") fortolkes sådan, at fungerende software er vigtigere for en kunde end omfattende dokumentation. Udviklere vil måske bruge værdien som undskyldning for at kode løs og undlade at dokumentere. Alene er Manifestet altså ikke specielt medgørligt og det uddyber heller ikke, hvordan værdierne overholdes. Her kommer de agile principper nærmere en afklaring og en realisering.

Manifesto for Agile Software Development

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

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

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

Figur 3 – Det Agile Manifest (www.agilemanifesto.org)

C.2.2 Agile Principper

De agile principper ses på Figur C.2.2. Der er i alt 12 principper, og de understøtter de 4 agile værdier i manifestet. For eksempel, understøtter princip nr. 3 den agile værdi nr. 2, idet princippet bl.a. udtrykker, at software og altså ikke dokumentation skal afleveres tidligt og hyppigt. Det vil sige, at følges princip nr. 3, så støttes den agile værdi nr. 2. Men principperne er dog stadig for abstrakte til at kunne omsættes direkte til praktisk brug. Det er her metoder og praktikker kommer ind i billedet.

Principles behind the Agile Manifesto

We follow these principles:

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

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

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

4) Business people and developers must work together daily throughout the project.

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

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

7) Working software is the primary measure of progress.

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

9) Continuous attention to technical excellence and good design enhances agility.

10) Simplicity--the art of maximizing the amount of work not done--is essential.

11) The best architectures, requirements, and designs emerge from self-organizing teams.

12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Figur 4 – De 12 Agile Principper (www.agilemanifesto.org)

C.2.3 Agile Metoder

Agile metoder kan betragtes som en konkretisering af de agile værdier og principper. Dvs. at værdier og principper operationaliseres til praktisk brug, idet de gøres specifikke. Metoder fortæller altså brugeren, hvilke skridt der skal tages for at gennemføre et princip. Disse metoder kan ofte dekomponeres til en række praktikker, der mere uniformt og i konkrete vendinger beskriver de enkelte skridt.

C.2.4 Agile Praktikker

Kort sagt er praktikker en detaljeret beskrivelse af, hvordan et eller flere principper realiseres. For eksempel, i XP gennemføres princip nr. 3 gennem praktikken "Weekly Cycle". Kort sagt siger praktikken, at en iteration (identificering af krav, design, konstruktion, test og kunde-feedback) strækker sig over en uge. Ved at følge praktikken, er man altså i stand til at aflevere små inkrementer af software til kunden tidligt og kontinuerligt (hver uge). Derved gennemføres princip nr. 3 og værdien i manifestets 2. sætning overholdes.

C.3 Hvad Kan Man Så Bruge Det Til?

Hvis metoder og praktikker er den faktiske realisering af de agile principper og værdier, hvorfor er det så vigtigt at beskæftige sig med dem? Jo, netop fordi praktikkerne er konkrete, vil det ofte være nødvendigt, at tilpasse dem til det miljø og den situation de skal virke i. Som eksempel vil praktikken "Weekly Cycle" ikke være så fornuftig at følge stringent, hvis udviklerne i forsøget på at nå deadlines på en uge, bliver overanstrengte og dermed svækker kvaliteten af softwaren. I det tilfælde vil princip nr. 8 og 9 blive brudt. Her vil det måske være fornuftigt at øge størrelsen på en iteration fra en uge til 14 eller 30 dage. Praktikken vil stadig realisere princip 3, men udviklerne kan nu bedre følge med, hvorved princip 8 og 9 også gennemføres. For at forstå Agil Softwareudvikling er man derfor nød til at forstå principperne, hvordan de understøtter de 4 agile værdier og sammenhængen imellem dem. Hermed kan man tilpasse praktikker og metoder til den pågældende situation og stadig udføre Agil Softwareudvikling.