UNIVERSITEIT GENT

FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

ACADEMIEJAAR 2014 – 2015

The Importance and Presence of Agile Principles In Agile Software Development

Masterproef voorgedragen tot het bekomen van de graad van

Master of Science in de Handelswetenschappen

Aygun Shafagatova

onder leiding van

Promotor: Prof. Dr. Manu De Backer
Commissaris: Tom Pauwaert

UNIVERSITEIT GENT

FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

UNIVERSITEIT GENT

FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

ACADEMIEJAAR 2014 – 2015

The Importance and Presence of Agile Principles In Agile Software Development

Masterproef voorgedragen tot het bekomen van de graad van

Master of Science in de Handelswetenschappen

Aygun Shafagatova

onder leiding van

Promotor: Prof. Dr. Manu De Backer
Commissaris: Tom Pauwaert

PERMISSION

SAMENTWATTING

De afgelopen jaren is de populariteit van agile softwareontwikkelingsmethoden enorm toegenomen. De ideeën, waarden en praktijken die agile softwareontwikkeling omvat hebben een grote invloed op de manier waarop softwareontwikkeling in de praktijk gebeurt. Hoewel er reeds behoorlijk wat onderzoek gedaan is naar agile ontwikkelingsmethoden en praktijken in het algemeen, is er nog relatief weinig aandacht besteed aan het belang van waarden en principes en hun aanwezigheidsgraad in agile ontwikkelingsprocessen. In deze context komen enkele vragen naar boven. Zijn de waarden en principes die in het Agile Manifesto vermeld staan bijvoorbeeld nog steeds in dezelfde mate aanwezig in agile softwareontwikkeling en zijn deze nog steeds belangrijk? In dit opzicht ontbreken empirische bewijsmiddelen en academische analyses in mainstream onderzoek.

Dit onderzoek heeft daarom als doel een volledig begrip te krijgen van de percepties en de werkelijkheid over de aanwezigheid van agile principes in agile ontwikkelingsprocessen. Het uiteindelijke doel van dit onderzoek is om een empirisch bewijs te leveren dat de originele ideeën en bouwstenen van de agile methode nog steeds gevolgd worden en om aan te tonen in welke mate beoefenaars voldoen aan deze waarden en principes. Om dit doel te bereiken, werd in dit onderzoek een enquête en een mapping gebruikt als onderzoeksmethodes voor het verzamelen van gegevens. De online- enquête werd uitgevoerd om te bepalen in welke mate agile praktijken gebruikt worden onder beoefenaars, en in welke mate zij belang hechten aan de agile principes. Verder is er een mapping gemaakt door agile deskundigen van academische en industriële achtergrond om te achterhalen in welke mate een agile praktijk welke agile principes ondersteund.

Uit de resultaten van het onderzoek blijkt dat er een aantal belangrijke verschillen bestaan in de mate van implementatie in de praktijk en in de belangrijkheidsgraad van de verschillende principes. Uit de analyse van de empirische gegevens blijkt dat er mogelijke inconsistenties zouden kunnen bestaan in het belang van de principes en de aanwezigheidsgraden, aangezien de naleving van de principes die als belangrijk beschouwd wordt zich op een verschillend niveau blijkt te bevinden. De studie wordt afgesloten met een aantal speculatieve opmerkingen over mogelijke verklaringen voor deze resultaten.

Trefwoorden: *Agile softwareontwikkeling, agile principes, agile praktijken, implementatie, het belang, de aanwezigheid.*

ABSTRACT

Agile software development methods have gained considerable popularity in recent years. It is widely recognized that, the ideas, values and practices which compose agile software development have had a major impact on how software development is done in practice in the past years. While there is a fair amount of research related to agile development methods and practices in general, relatively little attention has been paid to the importance of values and principles behind it, and their presence degree in agile development process. In this context questions arise, whether, after about 15 years, the values and principles stated in Agile Manifesto are still important and present in agile software development, and more specifically if at the same degree. In this regard, empirical evidence and academic analysis to date is lacking in mainstream research.

This study therefore aims to gain understanding on perceptions and realities about importance and presence of agile principles, particularly by examining the implementation degree of agile practices, importance degree of agile principles and by trying to lay down a possible relationship map between principles and practices. Hence, this study seeks to fill a small and yet not insignificant gap in existing research. Eventually, its objective is to provide empirical evidence if the original ideas and building-stones of agile approach are still being followed and to which degree practitioners are grounded on those values and principles. For this end, the study employed survey and mapping as a research method for collecting data. The online survey was conducted to determine the implementation level of agile practices and importance degree of agile principles among practitioners. Besides, the mapping was done by agile experts and academicians to find out the possible correspondence of agile practices and agile principles in terms of their support to latter.

The main findings reveal a number of remarkable differences in practice implementation degree and in principle importance range. The analysis of empirical data indicates that possible inconsistencies might exist in principle importance and presence degrees, since the adherence to principles which is perceived to be important seems to be not in equal level. The paper concludes with some speculative remarks on possible dynamics behind such results.

Keywords: *Agile software development, agile principles, agile practices, implementation, importance, presence.*

ACKNOWLEDGEMENTS

First of all, I would like to thank to my supervisor prof. Manu De Backer; in particular to express my sincere gratitude to my co-supervisor Tom Pauwaert for his vision, valuable feedbacks, inspiration and encouragement. In the wave of appreciation, I would also like to mention prof. Kieran Conboy, Sammy Becaus, Dale Strickler, prof. Mark Goh and Bart Vermijlen, who, with their vast expertise and deep knowledge on agile software development, contributed valuable input to this study. I would like to express my deepest gratitude to them for their involvement, criticism, eye-opening comments and professionalism. Special thanks go to all the respondents who took their time and participated in the survey by providing valuable input. My heartfelt thanks go to my classmate Christopher Deseck for his quick proofreading and continuous support.

Last but not least, my special indebtedness goes to my family, to my soul partner Ramin for his limitless support, care and compassion and to my wonderful daughters for their love, laugh and joy throughout my study.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ASD	Agile Software Development
BDD	Behavior Driven Development
CA	Comparative Agility
CSF	Critical Success Factors
DSDM	Dynamic System Development Method
FDD	Feature Driven Development
IEEE	Institute of Electrical and Electronics Engineers
ISD	Information Systems Development
ISO	International Organization for Standardization
LSD	Lean Software Development
MIT	Massachusetts Institute of Technology
ROI	Return on Investment
RQ	Research Question
TDD	Test Driven Development
XP	Extreme Programming

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is most adaptable to change.

Charles Darwin.

Agile Software Development (ASD) emerged as a response to the traditional way of developing software which had long delivery duration, was not able to react to changes rapidly and effectively and was loaded with piles of documentation. Since the launch of Agile Manifesto[1], ASD became hype and was promoted as something revolutionizing. There are currently a bunch of studies directed at ASD in general and its different aspects.

As generally accepted, the whole agile concept is based on values and principles laid down in the Agile Manifesto by a group of keen agile practitioners. This implies that agile principles should be at the core of and starting point for all methods and practices. Studies on different issues related to ASD have greatly multiplied in recent years. However, there has been relatively little analysis of importance and presence of agile principles in ASD process. After 15 years of existence, very limited knowledge is available on if those principles are still important and actually present in agile processes. More particularly, which practice is based on/supports which principle and the perceptions on importance of agile principles are the questions that have drawn very little attention in academic research. It is vitally important to examine these issues in order to be able to make sound conclusions about the link between theory (principles) and practice. Therefore, this study seeks to fill a small and yet not insignificant gap in existing research. Eventually, the aim of the study is to try to shed light on perceptions and realities about importance of agile principles. Of particular importance is to know if all principles are evenly important for agile practitioners; how they percept it and how they implement them. For example, they can consider certain principles very important but however if looked at what they use in practice it might be the case that they actually implement other principles more deeply than the rest.

1.1. Structure of thesis

This thesis is structured as following: In **first chapter**, the background information and social relevancy of the issue is highlighted. Afterwards, in **second chapter**, review of relevant literature is conducted under the subsections of theoretical literature and practitioners' literature. **Third chapter**

[1] The detailed explanation of Agile Manifesto and principles follows shortly.

mainly deals with research methods and design by elaborating both each chosen method and the process of collecting data. **Fourth chapter** outlines the analysis and description of empirical findings, while discussion of the findings is presented in **fifth chapter**. Finally, the last **conclusion chapter** summarizes the thesis findings and discussion by revisiting the research questions and by laying down the limitations and possible future research opportunities.

1.2. Background – social relevancy

Software development processes are becoming increasingly complex, on-demand, and difficult to define due to intangible character of software and fast-changing technology. It is not surprising that the highest project failure rates occur in software development domain. According to the Standish Group survey of 2013, only 39% of software projects were successful, while the rest were either failed or challenged. However, if comparing to 2004 where success rate was only 29%, there is a remarkable improvement (The Standish Group, 2013). And as we will find out shortly from the below-mentioned surveys, Agile is one of the main reasons of this improvement.

In the good old days of software development dawn, everything was much simpler and straightforward: most projects were about the automation of existing systems and therefore it could be easily planned and defined up-front. However, as the technology evolved to become more complex, as internet become the main medium of communication and operation, and as software became more a sort of consumer product, it became ever more challenging to define everything up-front and manage the frequent changes during the project. On the other hand, fast evolving technology and business environment required shorter time-to-market delivery which put also pressure on software development process. As more organizations seek to gain competitive advantage through timely deployment of Internet-based services, developers were under increasing pressure to produce new or enhanced implementations quickly. (Turk, 2002)

Furthermore, if we add to this the new technology trends, such as mobile applications, internet of things and big data, it becomes clear that traditional plan-based software development methodologies (for ex. waterfall) are simply not able to face those modern-day challenges effectively due to its very inflexible and heavily structured processes. (HP, L.P, 2015; Highsmith, 2010; Smith, 2009)

Therefore, agile development methods emerged as a response to the inability of previous plan-driven approaches to handle rapidly changing environments (Highsmith, 2002). They were developed primarily to address the problem of developing software in "Internet time" (Turk, 2002). Agile methods are a reaction to traditional ways of developing software and acknowledge the “need for an alternative to documentation driven, heavyweight software development processes" (Cohen, 2004). As a result, agile was originally promoted as a "movement" and viewed as a challenger to entrenched practices like Waterfall. It was introduced to the software development world in late 90s and early 2000s and since then has become very popular in the industry because of its ability to handle a high degree of uncertainty and change in software development projects. Originating from the so-called 'light-weight' methods and promoted through the publication of the Agile Manifesto (2001), the agile methods family have become highly prevalent in recent years. The term 'Agile' was coined during the summit of Agile Alliance of 17 agile practitioners in 2001 which also produced Agile Manifesto and agile principles, the basic values and principles that stand for Agile methodologies as a connecting bottom line (2001). Since then, within the industry there is a remarkable shift towards agile development and away from traditional waterfall methods. As a turning point, it brought remarkable transformation and unprecedented change in software development field and become a mainstream within it. Agile methods are attractive to software companies since they promise shorter time-to-market, as well as higher flexibility to accommodate changes in the requirements and thereby increased ability to react to changing customer (market) needs. (Williams L. C., 2003)

If we compare its relatively young age of about 15 years to its traditional counterparts, the ever rising use and popularity of agile methodologies is indeed impressive. Driven by a belief among adopters that Agile development is more customer-centric and enhances team collaboration, Agile was slow to start and quick to dominate, with most growth occurring in just the past five years (HP, L.P, 2015). According to the survey of Forrester conducted in 2009, 35% of respondents stated that Agile most closely reflects their development process, with the number increasing to 45% if definition of Agile is expanded[2]. (West, 2010)

The most recent survey of VersionOne (agile software tool provider) reveals that Agile is an increasing tendency within industry: as stated in it, 95% of development organisations practiced agile and a total of **45%** of respondents worked in development organizations where the majority of their teams are agile[3] (VersionOne, 2015). Alternatively, only **5%** of respondents work in a completely traditional/non-agile development organization, while in 2009 this was **31%** (VersionOne, 2015). That indicates rapid rise of agile in recent years. What is more, the 2014 survey of HP on usage of Agile supports this by going further and declaring Agile as a new norm in today's software development industry. According to this survey, agile development methods are

[2] Source: Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009, Base: 1,298 IT professionals
[3] The 9th annual State of Agile survey was conducted between July and October, 2014; a total of 3,925 completed responses from a broad range of industries in the global software development community were collected and analysed.

embraced by a majority of development teams and projects, and pure Waterfall approaches are now in the minority. As Figure 1 displays, two-thirds of participants described their company as either “Pure Agile” or “Leaning toward Agile”, another 24 percent indicate that they use a “hybrid” approach, meaning that they incorporate at least some Agile solutions and principles into the management of their software development projects[4]. (HP, L.P, 2015)

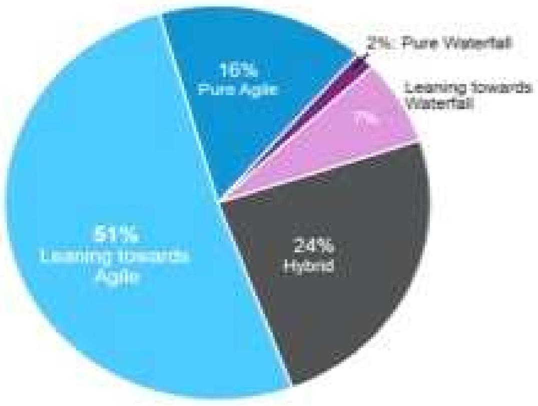


Figure 1: Primary development method used in organization across projects (HP, L.P, 2015)

These are considerably large numbers and demonstrate that agile do possess special approach and practices which addresses the challenges of software development. If further explored, the presented surveys provide insight on why agile is getting momentum so fast. According to VersionOne surveys, for already 4 years, top 3 benefits of adopting and actual improvements from implementing agile are:

1. Ability to manage changing priorities
2. Team productivity
3. Project visibility (2015)

Others include: Increased team morale/motivation; Better delivery predictability; Enhanced software quality; Faster time to market; Reduced project risk; Improved business/IT alignment (VersionOne, 2015). Alternatively, slightly different prioritisation of benefits is reported in HP survey; as Figure 2 illustrates, primary motivators for agile adoption are associated with improving team collaboration and increasing software quality and customer satisfaction. These factors, more so than efficiency gains, are the strongest benefits associated with Agile. (HP, L.P, 2015)

[4] 601 development and IT professionals participated in this survey of HP

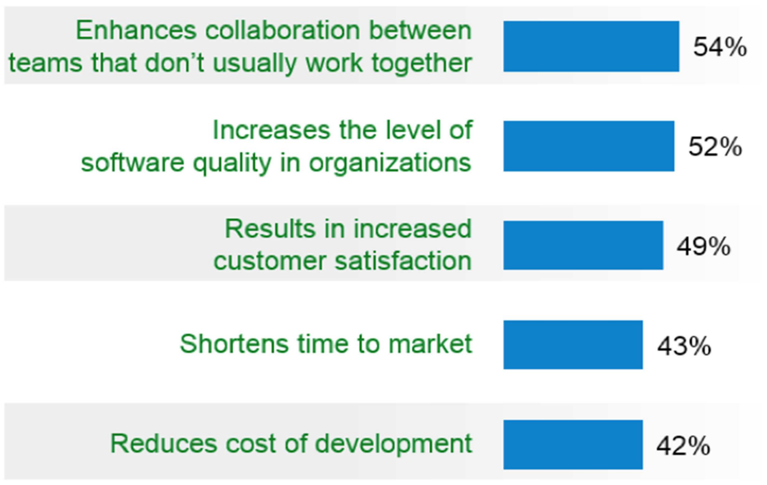


Figure 2: Percentage of respondents agreeing with statement about agile development (HP, L.P, 2015)

Moreover, according to the 2011 CHAOS report of the Standish Group, "software applications developed through the agile process have three times the success[5] rate of the traditional waterfall method and a much lower percentage of time and cost overruns." It is shown in Figure 3. The report goes on further to argue that, "The agile process is the universal remedy for software development project failure". (The Standish Group, 2013)

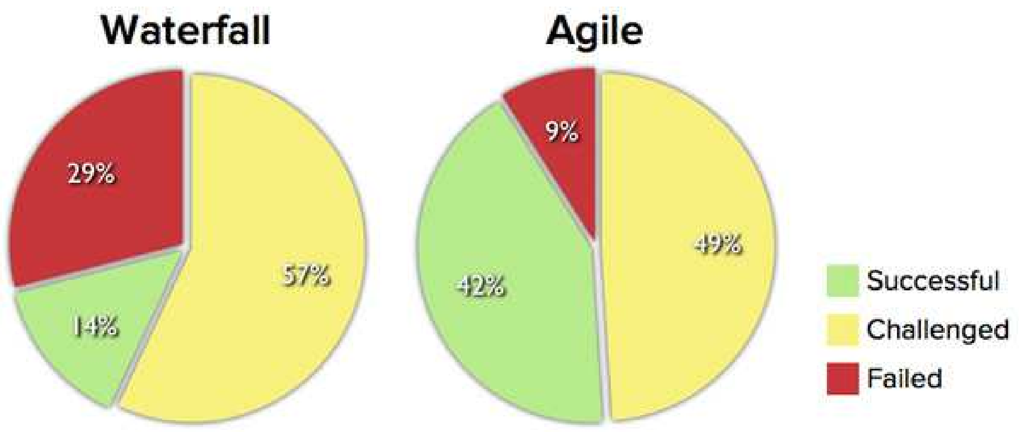


Figure 3: Success rates of Waterfall and agile projects (The Standish Group, 2013)

The aforementioned surveys and numbers provide support for the view that agile software development has had a huge impact on how software is developed worldwide. However, critics doubt whether the benefits of agile development outweigh the costs: The weaknesses of agile are

[5] The Standish Group defines project success as on time, on budget, and with all planned features.

listed as its code-focused approach, less documentation, chaotic appearance, limited applicability in certain situations such as in distributed teams or in very large and complex projects in large, hierarchical structured firms, lack of formal communication and not well-defined requirements (Dybå T. D., 2008; Turk, 2002; Meyer, 2014). Moreover, time and resources can be hard to estimate without a detailed plan in large organisations (Barlow, 2011). Another criticism of agile is that it limits the possibility to control and plan the content of a release (Boehm B. , 2002). Some proponents argue that agile can be sustained only with the teams that have high maturity and with customers that are willing to collaborate closely, thereby limiting its success chances[6].

Interestingly, as ASD methods become more popular, some view iterative, evolutionary, and incremental software development – a cornerstone of these methods – as the "modern" replacement of the waterfall model, however as Larman puts it, its practiced and published roots go back decades (2003). Although many view iterative and incremental development as a modern practice, its application dates as far back as the mid-1950s (Larman, 2003). Cohen also argues that most of the agile practices are nothing new. It is instead the focus and values behind Agile Methods that differentiate them from more traditional methods (Cohen, 2004). Boehm and Turner (2003) similarly noted that agile methods are incorrectly perceived as revolutionary and that most of the agile practices have origins in much older methods. Similarly, Cockburn and Highsmith, who are among founders of Agile Manifesto, explain that "what is new about Agile Methods is not the practices they use, but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and manoeuvrability" (Cockburn A. H., 2001). Practitioners agree that being Agile involves more than simply following guidelines that are supposed to make a project Agile. True agility is more than a collection of practices; it's a frame of mind. As Andrea Branca states, "other processes may *look* Agile, but they won't *feel* Agile" (Cohen, 2004). The focus of this study also lies on this exceptional aspect of ASD, put differently, the values and philosophical statements that makes agile distinct from other methods. It is of particular importance to investigate if agile practices used in the industry reflex those values and principles. Many argue that if they use some practices of agile mechanically, they become an agile team, but in reality they don't have basic understandings of agility and agile principles[7]. On this point, the possible correspondence of used agile practices and principles and the adherence degree to agile principles will be researched.

[6] Personal communication with agile experts
[7] Ibid.

2. LITERATURE REVIEW

Since ASD methods represent comparatively new approach to software development, relatively increasing attention has been paid to this topic. While reviewing the literature, distinction can be made between two categories: First, books and articles written by academicians, and the researchers of ASD; particularly this category of literature has been not so enormous and quite little academic research has been done on agile principles. Second, books, articles written by agile practitioners and experts from the industry that created, invented agile methodologies and use Agile in their work process for years. Due to practical oriented nature of ASD, works written by agile practitioners and pioneers are more popular and prevalent. Despite the increasing number of journal issues, conferences dedicated to ASD research, practitioners and consultants have largely driven the creation and dissemination of these methods and agile research has always lagged behind practice (Abrahamsson P. C., 2009). Fortunately, there has been a growing interest among researchers in studying ASD in recent years. In this study, those two categories of literature will be reviewed separately to get clear and both theoretical and practical insight on what have been done, written and researched.

2.1. Academic literature review

Since publication of Agile Manifesto, ASD has gained the attention of research community. Several distinct academic researchers have devoted considerable attention to studying agile systematically and thoroughly which have resulted in valuable works on ASD and on its different aspects. Hence, agile research has evolved from basic understanding of the methodologies to its success factors, to its scaling and adaptability capacities, and to post-adoption issues. The thorough review of research on ASD reveals 5 most important reports on introduction and overview of ASD - by Abrahamsson et al. (2002), Cohen et al. (2004), Erickson et al. (2005), Dybå and Dingsøyr (2008), and Dingsøyr et al (2012) - which contain the state-of-the-art and state-of-the-practice in terms of characteristics of the various agile methods and lessons learned from applying such methods in industry (Dingsøyr T. N., 2012). The following paragraphs briefly discuss some of these reports.

The first ever literature review of existing studies on ASD was done in a technical report by Abrahamsson et al in 2002. The report discusses the concept of agile development in general and presents processes, practices and roles of 10 agile methods by comparing them according to the development phases they support and developer competence they require. The report found scarce empirical support for the anecdotal claims on effectiveness of agile methods (Abrahamsson P. S., 2002). Although it presented a comparative perspective of agile methods, there is no clear definition of practices used by those methods and their relevancy to agile principles. Next, Cohen *et* *al.*'s review of 2004 focused on more origin aspects of agile - the history and roots of agile development, discussed the state-of-the-art with respect to the main agile methods, issues of project management in agile development. (Cohen, 2004). Cohen defined agile practices per method and analysed them according to their usage and support for certain phases of software development, management and communication by mapping them out with each other, which reflects the thesis study methodology. Still, he didn't analyse the correlation between practices and principles. On the other hand, Erickson et al. reported that a more well-established stream of research was found supporting XP practice of pair-programming (Erickson, 2005). Although the work of Erickson et al investigated the principles of XP that are being adopted by software industry professionals, they failed to address a broader set of agile principles.

Dybå and Dingsøyr, on the other hand, with their broader systematic review of 2008 aimed to determine what's known about agile development's benefits and limitations, the strength of the evidence supporting these findings, and the implications for the software industry and research community (Dybå T. D., 2008). They also revealed a lack of theoretical and methodological rigor (Dybå T. D., 2008). In general, this review presents a good overview and snap-shot of all agile researches done up to that date, whereas it did not cover the underlining relation between agile practices and principles; on the other hand, the usage, success factor and adoption of certain method practices are discussed in depth in this report.

Additionally, in 2012 Dingsøyr et al. provided excellent summary of literature to illustrate how research on agile has progressed in last 10 years by conducting a remarkable analysis of publications on ASD. They identified the most popular conferences ('International Conference on Agile Software Development' ("XP") based in Europe and 'Agile' in the US) and journals (IEEE Software, the Journal of Systems and Software, Information and Software Technology, and Empirical Software Engineering) in which publications on agile research appear. (Dingsøyr T. N., 2012) The information produced by the study is appreciated for getting an overview of agile publications and their categorization per different criteria. However, that doesn't provide any evidence on their relevance to the thesis study. Finally, the study indicated that the total number of publications shows that agile development has received much interest from the academic community; however, most of the research is inspired by practices emerging in industry rather than by academic theory (Dingsøyr T. N., 2012). Furthermore, Dingsøyr et al (2012) identified that in total from 2003 until 2011 five special issues and one special section on ASD have been published in scientific journals (Williams L. C., 2003; Siau, 2005; Abrahamsson P. C., 2009). Of particular importance is the issue edited by Abrahamsson which not only addressed the fundamental question of what constitutes ‘agility’ and agile methods, but also demonstrated approaches to broadening the scope of the applicability of agile concepts by addressing issues such as a better understanding of agile methods beyond adoption stage (Abrahamsson P. C., 2009). These concepts are highly relevant for the thesis study in order to define agility and to investigate the core values behind agile methods and the sustainability of agile adoption via the values. However, little evidence on implementation degree of broad range of practices and their adherence to agile principles is detected through those special issues.

Dingsøyr et al (2012) went then to argue that not enough attention is being paid to establishing theoretical underpinnings, when investigating agile development and its various practices, most research is practice-driven rather than theory-based. They urged agile researchers to embrace a more theory-based approach in the future when inquiring into these promising research areas of agile development (Dingsøyr T. N., 2012). The paper draws attention to theory-based approach by linking agile development to theory of coordination, to a theory of descriptive decision-making; attempt to develop a grounded theory of social factors in software development, and analyse the application of lean approaches in ASD (Dingsøyr T. N., 2012). Hence, this was one of the rare attempts to build a theoretical base for agile methods. On the other hand, this theory-driven approach is scattered and needs to be made coherent and systematic. What is more, it doesn’t make any linkage with agile principles and values. Furthermore, with respect to the kinds of agile methods that have been studied, 76% of the studies in the review by Dybå and Dingsøyr were done on XP. Studies on agility in general comes next, with 15% of the studies and very few on the Scrum development process, which was gaining significant traction in industry. (Dybå T. D., 2008)

No doubt, one of the most substantial academic works on ASD belongs to Dingsøyr et al, who have devoted considerable attention to investigating foundations and background of agile development, agile methods in practice, and principal challenges and new frontiers by accumulating several valuable research studies in one book (Dingsøyr T. D., 2010). Let us briefly review some of this research. Nerur et al. argues that the conceptual underpinnings of agile development are by no means novel and to this end outlined the principles of sociotechnical systems and explored intellectual foundation of agile methods (Dingsøyr T. D., 2010). Again, it presents a worthy attempt to theorize agile development process, however without referring to core agile principles. Abrahamsson et al., on the other hand, compared agile methods in an analytical framework taking it through the following lenses: project management support, life-cycle coverage, type of practical guidance, adaptability in actual use, type of research objectives and existence of empirical evidence for the method. They found out that different agile software development methods cover different phases of development life cycle, and most fail to provide project management support and offer little concrete guidance on how to use their solutions or adapt them. Moreover, Sharp&Robinson, based on their decade long studies of agile development teams, determined that story cards and Wall is crucial in supporting team collaboration, coordination, and communication (Dingsøyr T. D., 2010). This provides an excellent study which makes linkage between certain agile practices and principles, while limiting itself to collaboration and coordination. More technical and organisational issues are covered in the later sections: Livari describes organisational culture as a factor that affects the deployment of agile methods by suggesting that agile is incompatible with a hierarchical culture although agile development is more disciplined than ad hoc development. Next, Konboy et al. challenge the idea of the single customer involvement in ASD because of its narrow focus by arguing that current thinking regarding innovation in agile development needs to be extended to include multiple stakeholders outside the business unit. They go on to investigate applicability and implications of open innovation principles in agile development and advocate their integration to agile. (Dingsøyr T. D., 2010) It is worth noting that, while this presents some new perspectives and new principles into agile, it does not cover the existing principles. Moreover, the issues like challenges of scaling up agile while sticking in agile principles, situating agility in hierarchical environment indeed draw attention to the agile values and ideas; nevertheless they cover only very specific aspects which is out of scope of this study and therefore are not able to give the whole picture of perceptions and realities on agile principles.

The result of the review of the rest of academic literature can be clustered under the following categories based on their scope of research. They are congregated together since they address similar issues, definitions and problems faced in agile world and therefore fit to be in the same group of research.

First group of authors focused on management challenges of implementing agile in larger, non-agile environment (Boehm B. T., 2005), investigated how agile can be combined with plan-driven methods to create a hybrid methodology in large organisations (Barlow, 2011), possibilities of coexistence of agile and traditional development approaches next to each other (Vinekar, 2006), critically examined tailoring capabilities of agile methods (Conboy K. F., 2007) and discussed hybrid methodologies, focus on people, process, and technology dimensions of migrating to agile projects (Nerur, 2005). Similarly, Boehm&Turner identified some agile process and practices and discoursed challenges of implementing agile processes in traditional organisation by drawing attention to barriers like top-down structure of organizations on the way of scaling agile and give some valuable recommendations on how to overcome these barriers (Boehm B. T., 2005). Barlow et al. also constructed a possible theoretical framework and guideline on how it can be effective to implement agile development practices in large organizations by reviewing and analysing existing literature and theory (Barlow, 2011). While Boehm focused on practical issues, Barlow et al. presented a theoretical analysis and they both did not mention about the tailoring issues. On the other hand, Conboy examined tailoring aspect deeply and as he puts it, although agile from its name can be considered easily adaptable to unique organization environment; it is not evident that all agile methods have such tailoring capabilities (Conboy K. F., 2007). He goes on to explore tailoring challenges of agile in practice by focusing on specific critical success factors (CSFs) of tailoring, namely built-in contingency, clear rationale behind method practices, independence of method practices, and disciplined and educated tailoring of practices. Study concluded that most agile methods ignore those factors (Conboy K. F., 2007). Obviously, all this research is related more or less to agile adoption and adaptation issues and therefore can be classified in the same group. While these are definitely important problems, they have less of significance for the current research, only adherence to principles during adoption seems to be interesting research aspect, sadly it is not covered by these studies. The only authors that have investigated agile principles and practices adaptation issues are Vilain&Martins, who conducted case study which suggested that due to certain constraints, it is logical to neglect some of the agile principles and practices and yet keep success and the agility of software development (Vilain, 2011). Although this case study indicates some interesting insights on above mentioned problems, it is limited only to one company and to Scrum method making its validity questionable.

Second bunch of research articles studied the success factors of agile in practice. (Misra, 2009; Chow, 2008; Salo, 2004) Misra et al. identified success factors of agile software development by grouping them under organisational, people, and technical factors (Misra, 2009). Similarly, focus of Chow and Cao's research survey was critical success factors of agile across globe. The result identified three critical success factors for ASD projects: (a) Delivery Strategy, (b) Agile Software Engineering Techniques, and (c) Team Capability (Chow, 2008). Salo and Abrahamsson tried to evaluate and empirically validate the XP practices by conducting controlled case study in close-to-industry settings (Salo, 2004). There are some other research and case studies trying to correlate agile principles and success of ASD (Bermejo, 2014) but they remain in minority and only focus on achievement of success.

Finally, third group of studies presented mainly the critical analysis of ASD, identified its limitations and weaknesses (Conboy K. , 2009; Turk, 2002; Meyer, 2014). Research carried out by Turk et al. highlighted critical approach to agile by identifying its limitations in their article based on a study of principles and assumptions underlying a subset of the processes that claim to be “agile” by arguing that the same assumptions create limitations for agile development (Turk, 2002). Conboy went further and made a critical analysis of agile development by reconsidering the concept of agility as an Information System Development (ISD) concept independent from the development method or application area. He challenged agile methods and the associated literature by elaborating significant conceptual shortcomings of them in its current state, including a lack of clarity, theoretical glue, parsimony, limited applicability, and naivety regarding the evolution of the concept of agility in fields outside systems development (Conboy K. , 2009). Equally interesting is the in-depth introductory and critical analysis of ASD presented by Meyer (2014) who criticised the agile principles and practices and categorised practices as being the good, the hype and the ugly. Consequently, some valuable concepts related to agility, assumptions hidden in agile principles, and critical approach to agile principles and practices could be detected and it will be used during the definition and analysis phase.

Of particular importance and relevance to the thesis study was the research of Williams (2010; 2014). She has conducted several surveys and studies on agile principles and practices, their importance degree and implementation range. She even attempted to reconstruct agile principles arguing that the circumstances of 2001 when the principles were agreed, is not the same as current ones, therefore needs an improvement. (Williams L. D., 2014) All those indicate high relevancy of her studies for this research, therefore it will be discussed more in detail in the next chapters.

The aforementioned review suggests that agile software development research has gained momentum with some dedicated substantial reports, special issues of journals and books. Nevertheless, as Ågerfalk and Fitzgerald put it, “practice is ahead of research” in this area (Ågerfalk, 2006). Rajlich further argued that ASD brings a host of new topics into software engineering research, and that there exists a “backlog of research tasks” (Rajlich, 2006). Furthermore, most research is based on empirical studies and practices emerged from industry rather than on substantial theories. There is a lack of conceptual studies on agility and other issues as a universal concept and theoretical model. Nevertheless, some studies could be elaborated on certain agile practices and principles, on certain agile method practices and their adoption and success rates, on some assumptions and limitations coming from principles, and on criticism of agile principles and practices. Much research to date has focused on agile practices, while relatively little has been written about importance and presence of agile principles in development process and no correlation of principles and practices have been made for this end. Hence, there exist definitely a significant gap that deals with the studying the adherence degree to agile principles by examining the implementation rate of agile practices and perceptions of importance of agile principles and by mapping those practices and principles in order to make decent conclusions on describing the reality. Agile principles and practices are studied well separately, although it concerns only certain method practices. But their correlation study, particularly presence of agile principles in ASD is missing in this list. To date, this issue has not been explored in academia. Therefore, this study aims to fill this gap and to gain understanding of above mentioned issues of principle-practice correspondence and adherence to agile principles. It is expected to deliver discovering contribution to existing literature by focusing on the core ideas and principles of agile and by examining if those principles still are important for ASD process and are equally present in this process. It will once again lay emphasis on the importance of agile principles and values, whereby most agile research is concerned with agile practices.

2.2. Practitioners' literature review

Agile software development has grounded itself on real practices, thus it is not wonder that practitioners and consultants have largely driven the creation and dissemination of these methods. There exist a large number of different descriptive, guideline-like books, and articles on different agile methodologies and practices which are mostly written by the founders of certain methodologies. As a consequence, all those sources are practice-oriented and concerned more with how to become agile and what is agile. In this respect, Cockburn and Highsmith's breakthrough article on agile remains one of the earliest sources on agile which documented basic principles and practices of agile in short and became a starting point for agile development process (Cockburn A. H., 2001), so which is also important for the purpose of this study. Cockburn's another article written with Williams discussed some core understandings and evolution of agile discourse over time. Perhaps more interesting in this work is the explanation of defined vs. empirical processes for getting a real insight on specific character of software development process and the role of agile in it. (Williams L. C., 2003)

Next, Martin and Beck elaborated agile and its principles, particularly focusing on XP practices with specific coding examples (Beck, 2005; Martin, 2014), while Highsmith's book "Agile Project Management" is primarily concerned with project management aspects of agile and presents an alternative project management approach to traditional one (Highsmith, 2010). Moreover, Ambler defined and described all details on agile modelling[8] in his book and is considered foundation sources for agile modelling and XP practices (Ambler S. , 2002). "Scrum Guide" by its founders

[8] an agile method

Schwaber and Sutherland is an ultimate guide which lays down all Scrum processes, actors, roles and artefacts in plain language (Schwaber, 2013). On the other hand, Medinilla analysed agile from a management and leadership perspective, by taking it from the development team environment to the whole system (Medinilla, 2012). Consequently, all these and countless other practical literature (AgileAlliance; Coad, 1999; Cockburn A. , 2001; Stapleton, 1997; Poppendieck, 2001; Smith, 2009) has covered different aspects of agile development: some of them are general descriptions of certain method practices and others are more specific focusing on certain practices or environments. Likewise, it also proves the fact that like most previous methods, the development and promotion of these methods have been almost entirely driven by practitioners and consultants, with little participation from the research community during the early stages of evolution (Conboy K. , 2009). Therefore, much work has still to be undertaken to bring coherence to the current discourse on agility. Despite the copious research on agile software development and its ramifications, one cannot help but sense a lack of a unified framework that brings coherence to the seemingly disparate streams of research being pursued. (Dingsøyr T. N., 2012)

To sum up, most of the practitioners' literature is a valuable source as a starting point and as first-hand information on certain practices and methods which is mainly practical-oriented and deals largely with how to apply all those practices. However, some of them also state the importance of agile ideas and values by branding agile as a mind-set and attitude. From here comes the basic starting point of this research study – re-focusing on agile basic values and principles and going 'back to basics' in order to catch the purity of agile development. While there can be found much similar definitional literature on principles and practices separately, there has been little analysis of their correlation and of importance of principles and degree of usage of practices. Although equipped with more agile knowledge after reviewing all academic and practitioners' literature, it is still unknown how to relate the agile principles and practices correctly, and in which degree are those principles present in agile process after the 15 years of lifespan. Even if it is generally agreed that agile principles are important for ASD, its degree of importance and actual presence needs to be studied further.

2.3. Research questions

The aim of this master thesis is to contribute to understanding the importance and presence of agile practice in ASD process, and more specifically, by trying to lay down a possible relationship map between principles (concepts) and practices, it aims to answer the following research questions (RQ):

RQ1. What is the implementation degree of agile practices among practitioners?

RQ2. How important are agile principles for practitioners?

RQ3. What kind of correlation exists between agile practices and principles? In which degree do agile practices support/based on agile principles?

RQ4. Are agile principles, which are perceived to be important, also with the same degree actually present in agile software development process of practitioners through supporting practices?

With the help of the chosen research methodology, the aim of this study is therefore to explore these key research questions. With these questions, study seeks to elaborate the mind-set importance and presence in ASD. In this regard, it could be as double-checking control analysis to provide empirical evidence if the original ideas and building-stones of agile approach are still being followed and to which degree practitioners are grounded on those values and principles. Hence, this study can be considered as a 'back-to-basics' attempt in order to reflect all those agile hustle that is going around nowadays.

2.4. Conceptualisation of agile software development process

After reviewing the work of both academicians and practitioners and getting acquainted with mainstream research, we are in a better place to identify agility, agile principles, methods and main understandings.

2.4.1. Defining agility

While discussing ASD, first the concept ‘agility’, which stands at the core of those methodologies, should be identified and understood. What does it mean to be agile? What is agility?

Reviewing the existing literature shows that the current body of agile method knowledge suffers from a lack of clarity as to what constitutes agility. Hardly any two agile method texts or papers adopt the same definition of agility or agile method. It seems almost every piece of research adopts a unique interpretation of agility and it has been used by many different people to refer to very different phenomena (Conboy K. , 2009). However, according to Abrahamsson, this is to be expected to some degree (Abrahamsson P. C., 2009); Lyytinen & Rose (2006) argue that, in the context of ISD, agility as a concept needs to be multifaceted and contextual, and that agility is achieved through various different means depending on the project environment.

In this regard, if it is viewed from pure dictionary perspective, agility is often associated with such related concepts as nimbleness, suppleness, quickness, dexterity, liveliness, or alertness. Erickson et al. underline the importance of lightweight processes in agile development, defining agility as to ‘strip away as much of the heaviness, commonly associated with traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, and accelerated project deadlines’. (Erickson, 2005)

Equally interesting is the Cockburn’s claim that at its core, agility entails ability to rapidly and flexibly create and respond to change in the business and technical domains (Cockburn A. H., 2001). In essence, these ideas suggest a “light’ methodology that promotes manoeuvrability and speed of response” and ‘lightness and leanness’ (i.e., having minimal formal processes) ((Cockburn A. , 2007) in (Dingsøyr T. N., 2012)). More team based definition of agility is elaborated by Turk et al.; as they put it, the agility of a process is determined by the degree to which a project team can dynamically adapt the process based on changes in the environment and the collective experiences of the developers (Turk, 2002). While Turk stated that Agile Manifesto principles are the attempt to define what agility means, Dingsøyr had opposite opinion by arguing that the principles are not a formal definition of agility, but are rather guidelines for delivering high-quality software in an agile manner.

Interestingly, Highsmith has defined agility in 3 different contexts: First, he argues that agility is all about trusting in one's ability to respond to unpredictable events more than trusting in one's ability to plan ahead for them. Highsmith states that being agile means being able to “Deliver quickly. Change quickly. Change often” (Fowler, 2001). Second definition is made in project management context: ‘agility is the ability to both create and respond to change in order to profit in a turbulent business environment’ (Highsmith, 2010) and the last one is more universal in character assuming that ‘agility is the ability to balance flexibility and stability’ (Highsmith, 2002). Moreover, Highsmith argued that some mistakenly assume that agility connotes a lack of structure, but the absence of structure/stability generates chaos. Conversely, too much structure generates rigidity, so process centric methods fail to balance to create innovation (Highsmith, 2010).

Obviously, Conboy provided by far the most comprehensive definition of software development agility by systematically examining its various facets and definitions from related disciplines. By this he wanted to produce the definition of agility as a universal concept that is methodology independent and applicable for all of them. For this purpose, he made distinction between agility, flexibility, and leanness—in fact, agility is conceptualized to include and go beyond both flexibility and leanness. Hence, according to Conboy, agility is the continual readiness of software development method “to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment.” (Conboy K. , 2009)

Put differently, it could be concluded that agility is all about: creating and responding to unpredictable change rapidly and effectively; flexibility; lightness; short delivery of working software; collaboration; feedback, and focusing on customer needs and value. Hitherto, this covers most aspects of agility and could provide a basis for further analysis.

2.4.2. Defining the process: Agile software development

The phrase ‘agile software development’, plays actually a role of umbrella for different agile methods with distinct and sometimes conflicting practices. However, they also share many common characters that are defined by different practitioners and academic researchers. However, different authors mention different characters and sometimes define the same character in different terms. Central to this discourse is the argument that ASD is an incremental and iterative type of software development. Furthermore, agile development approaches can be defined as a development process which focuses on the client’s ever changing needs and responding to those changes in an effective and efficient manner. Agile ranks the client or the customer as the most important asset and delivers the product from this perspective by focusing on needs of customer. (Cockburn A. H., 2001; Dingsøyr T. N., 2012; Turk, 2002)

Equally interesting is the claim of Williams&Cockburn (2003) who state that ASD 'is about feedback and change' while they provide a valuable insight on fundamentals of software development process which makes it easier to see the whole picture. As they put it, in engineering, processes are classified as defined or empirical; a defined process is one that can be started and allowed to run to completion, producing the same results every time. However, software development cannot be considered a defined process because too much change occurs during the time that the team is developing the product. It is highly unlikely that any set of predefined steps will lead to a desirable, predictable outcome, because requirements change, technology changes, people are added and taken off the team, and so on. Therefore, Williams&Cockburn strongly advocate that software development should be considered as an empirical process which necessitates short "inspect-and-adapt" cycles and frequent, short feedback loops (2003). What is more, agile methods recognize this empirical (non-linear) character of software development process and adapt itself to the process whereby short inspect-and-adapt cycles help agile methodologies better handle the software industry's conflicting and unpredictable demands (Williams L. C., 2003). Likewise, Smith also defends this view by arguing that software development is definitely an empirical process, not a defined process. As he states, the problem is that 'we have been approaching software development for years as a defined process – and that approach doesn't work.' (Smith, 2009)

Boehm further elaborated the ASD process by defining it as lightweight processes that employ short iterative cycles, actively involve users to establish, prioritize, and verify requirements, and rely on a team's tacit knowledge as opposed to documentation. According to him, a truly agile method must be **iterative** (take several cycles to complete), **incremental** (not deliver the entire product at once), **self-organizing** (teams determine/decide the best way to handle work through informal communication and frequent, short meetings rather than relying on one owner to guide the project), and **emergent** (requirements emerge during the course of the project; processes, principles, and work structures are recognized during the project rather than predetermined). (Boehm B. T., 2005)

Similarly, Abrahamsson et al answered the question 'what makes the development method an agile one?' with following definitions: when software development is **incremental** (small software releases, with rapid cycles), **cooperative** (customer and developers working constantly together with close communication), **straightforward** (the method itself is easy to learn, to modify, well-documented), and **adaptive** (able to make last moment changes). (Abrahamsson P. S., 2002)

As a matter of fact, if summed up in a nutshell, ASD is an intensely iterative process, in which the entire project is broken up into several small projects, meaning that teams analyse, design, and code rigorously in short intervals, each lasting between one and six weeks (Larman, 2003). Each iteration deals with only a few prioritized features and ends up with a working system as a deliverable. The end of each iteration provides an opportunity to meet with the customer to evaluate the progress and get feedback; and start the cycle again. Developers and users dynamically prioritize features at the beginning of each iteration, and the project grows through evolutionary development. (Vinekar, 2006)

Next, Cohen defines common characteristics of agile methodologies adding more dimensions, which includes **iterative** development (allows the development team to adapt quickly to changing requirements) and a focus on interaction, **communication** (teams can make decisions and act on them immediately, rather than wait on correspondence), and the **reduction of resource-intensive intermediate artefacts** that do not add value (more resources can be devoted to the development of the product itself and it can be completed sooner) (Cohen, 2004). As Cohen states, "A great deal of the agile movement is about what I would call 'programmer power'" (Cohen, 2004). Interestingly, it is not coincidence that Forrester survey also found out that, many developers who have shied away from formal development methods in the past – believing them to be the province of "management" – have embraced Agile as a "formal" development process, because agile methods encourage more-collaborative development than do traditional approaches (West, 2010). All these characteristics add manoeuvrability to the process, whereby an agile project can identify and respond to changes more quickly than a project using a traditional approach. (Cohen, 2004)

It is worth noting that while most of the characteristics of ASD defined by different authors match and look similar, the definition of each characteristic seems to be not the same at all. Therefore, it is difficult to detect a coherent and clear definition.

Apparently, ASD has been characterised differently than plan-based or traditional development methods, mainly with the focus adapting to and embracing the change and delivering products of high quality through simple work-processes. Agile and traditional methods diverge on number of aspects, including their fundamental assumptions, approach to change and control, management style, knowledge management, role assignment, role of the customer, project cycle, development model and desired organisational structure (Dingsøyr T. N., 2012). In this regard, agile methods is viewed as a reaction to plan-based or traditional methods, which emphasize a "rationalized, engineering-based approach," incorporating extensive planning, codified processes, and rigorous reuse. In contrast, agile methods address the challenge of an unpredictable world, emphasizing the value competent people and their relationships bring to software development. (Dybå T. D., 2009) Table 1 perfectly summarizes these differences.

Dimensions/aspects	Traditional view	Agile perspective
Design process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness, ability to better respond to changing customer requirements quickly
Problem-solving process	Selection of the best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/generative
Key characteristics	Control and direction Avoids conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication; integrates different worldviews Embraces conflict and dialectics Encourages exploration and creativity; opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical and/or philosophical roots	Logical positivism, scientific method	Action learning, John Dewey's pragmatism, phenomenology
processes	defined	empirical
Rules	Inclusive	Generative
Life-cycle	Follows development phases for all features at the same time	Follows all development phases per few features in short iterations

Table 1: Traditional and agile perspectives on software development[9] (Dybå T. D., 2009)

[9] last two aspects are added by the author believing that they cover important difference dimensions

2.4.3. Agile values and principles

Agile values and principles constitute the core concept of ASD and are also the focus of this thesis research. It is vitally important to understand the main driving ideas behind agile methods and practices in order to be able to make sound conclusions and some accurate correlations between those practices and principles. While agile can be seen just as an another software development process, but as Smith puts it, there is a lot more to Agile than just a process or just a set of practices; agility is more of a mind-set – a way of thinking about software development. He then goes on to argue that agile mind-set can be applied to any process using any set of practices. (Smith, 2009) Smith has illustrated the understanding of agile in figure 4 as follows:

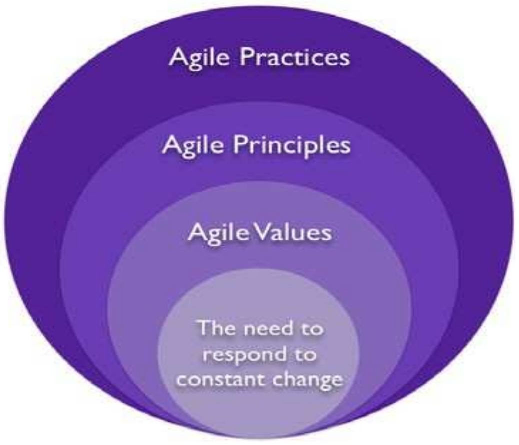


Figure 4: The relationship between agile values, principles, and practices (Smith, 2009)

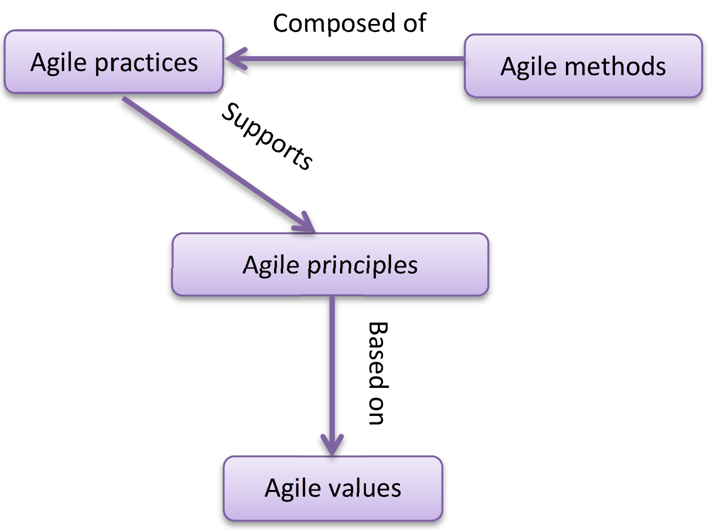


Figure 5: Relationship diagram of Agile values, principles, practices and methods (developed by the author)

Similarly, designing a relationship diagram of agile components also presents a valuable illustration. As figure 5 implies, the principles follow from the values; the practices, roles and artefacts follow from the principles. Hence, reading the Agile Manifesto is enough to see that "Agile" is not just a collection of software techniques but a movement, an ideology, a cause (Meyer, 2014). Particularly this ideology, mind-set aspect of ASD is what serves as a starting point and basis for this study, by trying to find out if practices are correlated with this mind-set and if practitioners find this mind-set important for their work process or they simply implement the practices without paying attention to core agile ideology and mind-set and without adhering to agile principles.

Agile Manifesto (short for: Manifesto for Agile Software Development), which was agreed during the meeting of 17 industry experts in 2001[10], presents strong value statement that yield the most benefit to a software development process and became a bottom-line conceptual basis for all agile development methods, consisting of 4 values and 12 principles. While each of those practitioners had different background, founded completely different agile methodologies with distinct practices, they were able to agree on issues for encouraging better ways of developing software (Ambler S. , 2002). Meyer defines values as 'general assumptions framing the agile view of the world" (Meyer, 2014). The Manifesto for Agile Software Development reads as follows:

"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." (History: The Agile Manifesto, 2001)

The Manifesto is defined by four simple value statements – the important thing to understand is that while you should value the concepts on the right side, you should value the things on the left side even more. A good way to think about the manifesto is that it defines preferences, not alternatives, encouraging a focus on certain areas but not eliminating others (Ambler S. , 2002). In each bullet point, the first segment indicates a preference, while the latter segment describes an item that,

[10] Agile Alliance was also established at the same meeting.

though important, is of lesser priority. This distinction lies at the heart of agility, but simply asking people to list what's valuable doesn't flesh out essential differences. (Fowler, 2001)

Of particular importance is that, with the Manifesto, agile proponents aimed to *refocus* the attention within software development. Interestingly, the Alliance recognizes the importance of process and tools, with the additional recognition that the interaction of skilled individuals is of even greater importance. Similarly, comprehensive documentation is not necessarily bad, but the primary focus must remain on the final product - delivering working software. Contract negotiation, whether through an internal project charter or external legal contract, isn't a bad practice, just an insufficient one. Contracts and project charters may provide some boundary conditions within which the parties can work, but only through ongoing collaboration can a development team hope to understand and deliver what the client wants. (Fowler, 2001) In this regard, Highsmith&Fowler emphasise that “the agile movement is not anti-methodology; in fact many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modelling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely used tomes. We plan, but recognize the limits of planning in a turbulent environment.” (Fowler, 2001) Next, let us briefly review each value statement to get better understanding:

- **Individuals and interactions** over processes and tools.

This suggests that the ingenuity and competence of people as well as their interactions and collaborations are of greater value than tools and processes. There is a distinct move towards collaborative development, with people being accorded privileges over processes that formerly constrained them (Dingsøyr T. N., 2012). In other words, according to agilists, people are the most important ingredient of success[11]; a good process will not save a project from failure if the team doesn't have strong players, but a bad process can make even the strongest of players ineffective. For Martin, working well with others, communicating and interacting, team-player skills are more important than raw programming talent. What is more, building the team is more important that building the environment. He therefore advises to work to create the team, and then let the team configure the environment on the basis of need. (Martin, 2014)

- **Working software** over comprehensive documentation.

[11] Personal communication with agile practitioners

In ASD world, delivering a high-quality working system to the customer is more important than producing copious documentation (Vinekar, 2006). As a result, a dominant “lean” mentality was advocated with a view to minimizing unnecessary work, particularly with regard to the creation of wasteful documentation. While this was misconstrued by many to mean “no documentation”, the discerning realized that this meant documenting only what was absolutely necessary and nothing more. (Dingsøyr T. N., 2012) Martin argues that huge software documents take a great deal of time to produce and even more time to keep in sync with the code. Therefore, it is always a good idea for the team to write and maintain a short rationale and structure document (Martin, 2014). Consequently, as he further states: “that document needs to be short and salient. By *short*, I mean one or two dozen pages at most. By *salient*, I mean that it should discuss the overall design rationale and only the highest-level structures in the system” (Martin, 2014). Interestingly, he presents *Martin's First Law of Documentation* in his book: *‘Produce no document unless its need is immediate and significant’.* (Martin, 2014)

- **Customer collaboration** over contract negotiation.

The customer collaboration, which became a buzzword of modern business, is the focus of this value. Apparently, the active participation and constant involvement of the customer in systems development yields greater benefits than the fulfilment of predetermined requirements specified in a contract (Vinekar, 2006). As a result, customers/stakeholders are no longer just at the fringes of software development, but actively shape and guide the evolution of the end software product or service (Dingsøyr T. N., 2012). Furthermore, Martin believes that successful projects involve customer feedback on a regular and frequent basis; rather than depending on a contract, or a statement of work, the customer of the software works closely with the development team, providing frequent feedback on its efforts. He then goes on to argue that a contract that specifies the requirements, schedule, and cost of a project is fundamentally flawed. According to Martin, the best contracts are those that govern the way the development team and the customer will work together and gives an understanding of everyone’s rights and responsibilities. (Martin, 2014)

Put differently, customer collaboration means that all players—the sponsor, customer, user, and developer—are on the same team. Merging their different experiences and expertise with goodwill allows the combined group to change directions quickly so they can produce more appropriate results and less expensive designs. Contracts or project charters with the customers are necessary, but without collaboration, they are insufficient. (Cockburn A. H., 2001)

- **Responding to change** over following a plan.

From here follows that recognizing the inevitability of change and embracing it, rather than attempting to cope with it through extensive planning, provides the nimbleness needed to survive in a turbulent business world (Highsmith, Agile Project Managent: Creatinng Innovative Products, 2010). The focus, therefore, is on adaptation and innovation rather than prediction and control (Vinekar, 2006). In this regard, there was an acceptance of the fact that uncertainty was a part and parcel of software development, and that the inherent tendency to control variations through statistical and other means was futile (Dingsøyr T. N., 2012). Of particular importance is the provision that agile approach accepted change as a norm in software development rather than exception. In fact, the change was a driver factor for agile movement against traditional approach. According to Martin, the ability to respond to change often determines the success or failure of a software project. Since the course of a software project cannot be planned very far into the future, he recommends that plans should be flexible enough to adapt to changes in the business and technology and what's more, a better planning strategy is to make detailed plans for the next week, rough plans for the next 3 months, and extremely crude plans beyond that. (Martin, 2014)

To sum up values evaluation, as Highsmith states, it doesn't mean that processes, tools, documentations, plans, contracts are unimportant. In fact, there is tremendous difference between one thing being more or less important than another and being unimportant. Tools are critical to speeding development and reducing costs. Contracts are vital to initiating developer-customer relationships. Documentation aids communication. However, the items on the left are the most critical. Without skilled individuals, working products, close interactions with customers, and responsiveness to change, product delivery will be nearly impossible. (Highsmith, 2010)

The Manifesto is trying to point out that organizations traditionally put a huge emphasis on the items on the right, such as processes and tools, and neglect the items on the left, such as the interaction between individuals. In this respect, an Agile mind-set promotes the *re-emphasize* that an agile process can and sometime should contain some of the items on the right; but you need to make sure that each of those items adds indispensable value to the project. (Smith, 2009)

Interestingly, Meyer, on the other hand, has made his own list of agile values which he considers more substantial than that of Manifesto and read as follows:

1. Redefined roles for developers, managers and customers - Agile methods redefine and limit the manager's job by transferring many of the duties to the *team* as a whole, including the

selecting tasks to be performed and assigning them to developers. Most methods advocate including a customer representative in the development team itself.

2. No “Big Upfront” steps – that means no extensive planning at the beginning of the project.
3. Iterative development – short time-boxed development.
4. Limited, negotiated functionality – agile approach advocates limiting features to the most important ones, as measured by their business value: their ROI.
5. Focus on quality - understood as achieved through continuous testing. (Meyer, 2014)

Although Meyer’s values point out the substantial agile aspects, they are not much explicable and clear-formulated as its predecessors.

2.4.4. Agile Principles

Based on the Manifesto, Agile Alliance formulated a collection of principles that defines the criteria for ASD processes. Basically, agile principles form a foundation of common sense upon which one can base successful software development efforts (Ambler S. , 2002). Meyer defines principles as “core agile rules, organizational and technical” (Meyer, 2014). Similarly, principles are “*domain-specific guidelines for life*” (Beck, 2005) showing how the values can be applied in different areas. Furthermore, principles also act as a bridge over the gap between values and practices, bringing the values into concrete, specific application. Below are the 12 principles of agile software development with relevant discussion on each of them:

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

According to MIT Sloan Management review which analysed software development practices that have significant impact on the quality, *“The more frequent and earlier the deliveries, the higher the final quality*”[12]. An agile set of practices delivers early and often. As Martin puts it, ‘we strive to deliver a rudimentary system within the first few weeks of the start of the project. Thereafter, we strive to continue to deliver systems of increasing functionality every few weeks.’ (Martin, 2014) Likewise, Smith also believes that the earlier you can start delivering working software, the earlier you can begin satisfying your customer (Smith, 2009). This suggests delivering early gives possibility for quick wins and early feedback about the requirements, the team and the process. Furthermore, delivering frequently enables continuous wins for the team, fast feedback and up-to-date priorities. (Cockburn A. , 2007)

[12] Product Development Practices that work: How internet companies build software, MIT Sloan Management Review, 2001

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

Martin considers this as a statement of attitude and it is not surprising that most authors value this principle very highly as the main distinction from traditional development processes. According to Martin, the participants in an agile process are not afraid of change; on the contrary, they view changes to the requirements as *good* things, because those changes mean that the team has learned more about what it will take to satisfy the customer. Therefore, an agile team works very hard to keep the structure of its software flexible, so that when requirements change, the impact to the system is minimal. (Martin, 2014)

Smith further explains it from broader point of view: Since the growing unpredictability of the future is one of the most challenging aspects of the new economy, turbulence in both business and technology causes change, which can be viewed either as a threat to be guarded against or as an opportunity to be embraced. Rather than resist change, the agile approach strives to accommodate it as easily and efficiently as possible, while maintaining an awareness of its consequences. Although most people agree that feedback is important, they often ignore the fact that the result of accepted feedback is change. Agile methodologies harness this result, because their proponents understand that facilitating change is more effective than attempting to prevent it. (Smith, 2009)

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

Iterative incremental development is essential in ASD. However, one should keep in mind that deliver is not the same as release. According to Martin, agile teams deliver *working* software, and they deliver it early and often. Agile approach is not content with delivering bundles of documents or plans, for agile practitioners they are not counted as true deliveries. In fact, the ultimate goal is delivering software that satisfies the customer's needs. (Martin, 2014) Cockburn also considers that software delivery frequency should be as short as possible. It is up to customer, how often one can take in the new delivery for their review. If the frequency is months, it is suggested to use intermittent, light mid- reviews. (Cockburn A. , 2007)

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

Fowler&Highsmith defend the point of putting the phrase of "daily" in the principle by arguing that they want to emphasize the software customer's continuing commitment to actively take part in, and indeed take joint responsibility for, the software project (Fowler, 2001). Obviously, in order for a project to be agile, customers, developers, and stakeholders must have significant and frequent interaction. As Martin puts it, a software project is not like a fire-and-forget weapon, it must be continuously guided (Martin, 2014). Hence, daily cooperation with developers and business people or end user representatives is one success factor for the project. Whereas daily availability is not practical in most of the projects, stakeholders need to be available on short notice when a discussion is needed. (Cockburn A. , 2007)

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

Principles, patterns, and practices are important, but it's the people who make them work. As Cockburn advocates, all the tools, processes have second-order effect on the outcome of a project; people are who matters most in the end and has first-order effect (Cockburn A. , 2007). Thus, people are the most important success factor; all other factors, process, environment, management, and so on are second-order factors and are subject to change if they are having an adverse effect on the people (Martin, 2014). Moreover, Highsmith goes even further and argues that decisions must be made by the people who know the most about the situation (Highsmith, 2002). This means that managers must trust their staff to make the decisions about the things they're paid to know about (Fowler, 2001). Thus, ASD puts much importance on team empowerment and encourages involving the team in decision–making process. Since motivated individuals are keys to success in projects, management should provide the tool and training and support to get their work done and then keep out from their way to leave them space to express themselves. (Cockburn A. , 2007)

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

As Martin states, in an agile project, people *talk* to one another (Martin, 2014). The primary mode of communication is human interaction. Therefore, written documents are created and updated incrementally on the same schedule as the software and only as needed, so that documentation becomes a by-product of the process. Martin further argues that a lot of context is lost in communication over email and instant messaging by pointing out that ambiguity increases with non-verbal communication. (Martin, 2014) However, as most gets it wrongly, the distinction between agile and document-centric methodologies is not one of extensive documentation versus no documentation; rather, as Fowler claims, a differing concept of the blend of documentation and conversation required to elicit understanding. "The issue is not documentation, the issue is understanding!" (Fowler, 2001)

7. Working software is the primary measure of progress.

This principle of agile makes it possible to know much earlier if projects fail; it takes risks to earlier phase and gives opportunities to deal with them. Cockburn acknowledges that working software is the measure of progress because there's no other way of capturing the subtleties of the requirements, since it tells more about the situation than plans and documents. Obviously, working software has to have main functionalities, the user interface and the algorithms, so that both can be evaluated by customer and valuable feedback can be gathered. (Cockburn A. , 2007) Martin strengthens this argument by stating that agile projects measure their progress by measuring the amount of software that is currently meeting the customer's need. They don't measure their progress in terms of the phase they are in or by the volume of documentation that has been produced or by the amount of infrastructure code they have created. Put differently, they are 30 percent done when 30 percent of the necessary functionality is working. (Martin 2014)

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

This principle has 2 dimensions: one relates to social responsibility, the other to project effectiveness. Sustainable development means finding a working pace (40 or so hours a week) that the team can sustain over time and remain healthy (Fowler, 2001). The team does not take off at full speed and try to maintain that speed for the duration. Rather, it runs at a fast but sustainable pace (Martin, 2014). Thus, projects should be organised in a way that everyone involved would have reasonable working hours enabling them to stay alert, effective and engaged. (Cockburn A. H., 2001)

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

According to Glass, this principle is about continuous improvement of the software which mainly refers to maintenance that consists of fixing bugs and also refactoring (improving the code) (Glass, 2001). He states that this principle derives from programmer point of view. Agile approaches emphasize quality of design, because design quality is essential to maintaining agility. Project's design is enhanced continually throughout the project (Fowler, 2001). Martin believes that the way to go fast is to keep the software as clean and robust as possible. Thus, all agile team members are committed to producing only the highest quality code they can. They do not make messes and then tell themselves that they'll clean them up when they have more time, instead, they clean any messes as they are made (Martin, 2014). Hence, technical excellence is an essential enabler for a truly agile software development process.

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

In an agile project, it's particularly important to use simple approaches, because they're easier to change. Fowler believes that it's easier to add something to a process that's too simple than it is to take something away from a process that's too complicated. Hence, there's a strong taste of minimalism in all the agile methods. Giving people a simple set of rules and encouraging their creativity will produce far better outcomes than imposing complex, rigid regulations. (Fowler, 2001) Agile teams don't put a lot of importance on anticipating tomorrow's problems; nor do they try to defend against all of them today. Rather, they do the simplest and highest quality work today, confident that it will be easy to change if and when tomorrow's problems arise. (Martin, 2014) However, Glass warns that simplicity should not mean neglecting design by starting programming as soon as possible. (Glass, 2001)

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

Emergent properties (understood as innovation and creativity in human organizations) are best generated from self-organizing teams in which the interactions are high and the process rules are few (Fowler, 2001). There are 2 different approaches while explaining this principle. One is organisational, and the other is more technical. Martin states that an agile team is a self-organizing team implying that responsibilities are not handed or assigned to individual team members from the outside but rather are communicated to the team as a whole. Consequently, the team determines the best way to fulfil those responsibilities, which is why, agile team members work together on all aspects of the project and each member is allowed input into the whole. Interestingly, no single team member is solely responsible for the architecture or the requirements or the tests; instead the whole team shares those responsibilities, and each team member has influence over them. (Martin, 2014) Cockburn has different view of this principle, he stresses more the ability of the team to allow architecture, requirements and design to adjust over time, become better and more accurate gradually, and not fix them too accurately in the beginning (Cockburn A. , 2001). Hence, this principle supports mainly the design and analysis phase of development, and also organisational aspect.

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

This advocates that an agile team continually adjusts its organization, rules, conventions, relationships, and so on accordingly. In this respect, an agile team knows that its environment is continuously changing and knows that it must change with that environment to remain agile (Martin, 2014). Improving the team work habits regularly can be said to be agility in work methodology (Cockburn A. , 2007). According to Smith, the idea of always reflecting on what you're doing and trying to figure out better ways to do things is the essence of continuous improvement (Smith, 2009). So, any agile team must refine and reflect as it goes along, constantly improving its practices in its local circumstances. Hence, trust in people, believing that individual capability and group interaction are keys to success extends to trusting teams to monitor and improve their own development processes (Fowler, 2001). This principle concerns the maturity of agile team and its focus on improvement.

If summed up, according to the agile principles pronounced in the agile Manifesto, motivated and empowered software developers – relying on technical excellence and simple designs – create business value by delivering working software to users at regular short intervals. While individual principles and practices of agile development were not entirely new to the software community, the way in which they were put together into a cogent "theoretical and practical framework" was certainly novel (Williams L. C., 2003). Interestingly, some other interpretations and classifications of agile principles by different researchers are identified in the literature. For example, Fogelström compared 3 groups of agile principles:

AP1. Feature orientation: The main focus is on the production of features as soon as possible. The goal is to deliver working functionality that is perceived to have the most value for the customer, and this is done in small and frequent iterations. (Fogelström, 2010) Hence, this one includes in itself principles related to working software, satisfying the customer, and short delivery periods.

AP2. Reactive development: Reactive development is about responding to a change instead of planning ahead, and delaying decisions as long as possible. Agile methods promise flexibility and follow the ideology that one cannot control the world, thus the strategy is to respond to a change rather than plan for it. Examples of reactive practices are refactoring, adjusting requirements' priorities, and release scope after each iteration and delaying decisions for as long as possible. (Fogelström, 2010) Principles 2, 9 and 12 are mainly applicable here.

AP3. Evolving project (release) scope: Perhaps one of the main distinguishing features of agile development is the change from a fix-scope approach to a more open-ended approach. In the traditional fix-scope approach much effort is spent on defining and planning the content of a product release of a project upfront. In agile the release scope is emerging in the process of development rather than planned ahead. (Fogelström, 2010)

Alternatively, Meyer presented a critical approach to agile principles which is why he actually reformulated them. He developed own list of principles and congregated them into following categories:

Agile Principles	
Organizational	**Technical**
1. Put the customer at the centre.	6. Develop iteratively.
2. Let the team self-organize.	6.1. Produce frequent working iterations.
3. Work at a sustainable pace.	6.2. Freeze requirements during iterations.
4. Develop minimal software.	7. Treat tests as a key resource:
4.1. Produce minimal functionality.	7.1. Do not start any new development until all tests pass.
4.2. Produce only the product requested.	7.2. Test first.
4.3. Develop only code and tests.	8. Express requirements through scenarios.
5. Accept change.	

Table 2: Agile principles reconstructed by Meyer (Meyer, 2014)

As Table 2 indicates, those are not the principles listed in the Agile Manifesto. Meyer has criticised the original principles, by accusing them for being more like practices, platitudes and assertions rather than real principles. He further discusses the characteristics and definition of good principles. According to him, a good methodological principle is both *abstract* and *falsifiable*. Abstractness differentiates principles from *practices*; falsifiability distinguishes them from *platitudes* (Meyer, 2014). Furthermore, he goes further to argue that in emphasizing and popularizing these principles, the agile movement places itself in the best tradition of software engineering — of the very compendium of wisdom, accumulated over several decades, which it so haughtily deprecates (Meyer, 2014). It is worth noting that some of the Meyer's arguments are indeed logical and present a fresh look into the agile discourse. Obviously, some agile principles have ambiguous elements and are open for the discussion. However, for the purpose of this study, the principles of Agile Manifesto will serve as a framework, since they represent the ideas of mainstream agile concept founders who formulated them, and consequently are most popular and accepted in the industry. What's more, Meyer's principles are also based to original ones, while they are very new to the literature and little known in the industry.

Turk, on the other hand, analysed another aspect of principles by elaborating assumptions that are hidden as implicit statements within principles. Below is the list of assumptions with related principles (Turk, 2005):

- Principles 1, 4, 5 - Visibility Assumption, Iteration Assumption
- Principle 2 - Customer Interaction Assumption, Team Communication Assumption
- Principle 8 - Face-to-Face Assumption, Documentation Assumption
- Principle 3 - Changing Requirement Assumption, Cost of Change Assumption
- Principles 6,7,12 - Team Experience Assumption, Self-Evaluation Assumption, Self-Organization Assumption
- Principles 9, 10 - Quality Assurance Assumption, Continuous-Redesign Assumption
- Principle 11 - Application-Specific Development Assumption, Continuous-Redesign Assumption (re-iterated) (Turk, 2005)

He further explains all those assumptions and believes that those assumptions should be fulfilled in order agile to be successful in any environment (Turk, 2005). It is important to note that those are very interesting dimensions of principles, because it gives better understanding of principles via uncovering the 'for-granted taken' conditions that is required in agile process.

Interestingly, if carefully examined, it is easy to distinguish the principles' groups which concern similar issues. As a consequence, the following categorization of principles could be developed:

- Team/organizational related – Principles 5, 6, 8, 11, 12
- Working Software related – Principles 1, 3, 7,
- Collaboration/communication related – Principles 2, 4, 6, 12
- Quality and technical related – Principles 9, 10, 11, 12
- Customer satisfaction related – Principles 1, 2

It is not surprising that some principles fall under more than one group, since they cover multiple dimensions of ASD. Likewise, all above mentioned classifications of principles according to different criteria present useful insights on different dimensions of agile principles. However, only the last categorisation developed by the author will be used in the analysis phase of the study to further elaborate the principle/practice correlation and find out which category of principles are considered to be more important than the others and subsequently which category is more present than others.

2.4.5. Agile Practices

The above-mentioned principles have spawned a number of practices that are believed to deliver greater value to customers. In other words, to achieve the principles presented above, agile methods promote a set of practices. Meyer identifies agile practices as "specific activities practiced by agile teams" (Meyer, 2014). As a whole, agile practices are the collection of different practices that belongs to different agile methods. As the practices promoted to support agile values and principles vary with the method, each agile method has its own bunch of practices that covers certain agile issues. Put differently, agile practices define how agile methods are implemented in practice. At the core of these practices is the idea of self-organizing teams whose members are not only collocated but also work at a pace that sustains their creativity and productivity. The principles encourage practices that accommodate change in requirements at any stage of the development process. Furthermore, customers are actively involved in the development process, facilitating feedback and reflection that can lead to more satisfying outcomes. (Dingsøyr T. N., 2012)

According to Cockburn&Highsmith, a team isn't agile if the feedback loop with customers and management is six months. Agile approaches therefore recommend short iterations in the two- to six-week range during which the team makes constant trade-off decisions and adjusts to new information (Cockburn A. H., 2001). In their pioneering article, Highsmith&Cockburn (2001) presented a first initial overview of core agile practices and how they were used by different methods. They have chosen the following practices:

- **Feature planning and dynamic prioritization -** Agile approaches combine these short iterative cycles with feature planning and dynamic prioritization. XP uses story cards; Scrum uses the term "backlog"; ASD and Feature-Driven Development refer to features. The key point is that agile approaches plan features, not tasks, as the first priority, because features are what customers understand. Dynamic prioritization means that at the end of iteration, the customer can reprioritize the features desired in the next cycle, discarding originally planned features and adding new ones.

- **Feedback and change -** Because they are most applicable to turbulent, high-change environments, agile approaches recommend a variety of practices for constant feedback on technical decisions, customer requirements, and management constraints.
- **Focus on teamwork -** Team proximity and intense interaction between team members are hallmarks of all agile methods. Using agile development methods requires close customer partnerships. (Cockburn A. H., 2001)

Later on, Boehm (2005) described another key set of agile practices:

- **Embracing change**: Seeing change as an ally rather than an enemy. Change allows for more creativity and quicker value to the customer.
- **Fast cycles**, **frequent delivery**: Scheduling many releases with short time spans between them forces implementation of only the highest priority functions, delivers value to the customer quickly, and speeds requirements emergence. Time boxing, for example, establishes specific time frames that are then filled with as much prioritized functionality as can be developed.
- **Simple design**: Designing for the battle, not the war. The motto is YAGNI (You Aren't Going to Need It). The anti-motto is BDUF (Big Design Up Front). Strip designs down to cover just what you're developing. Since change is inevitable, planning for future functions is a waste of effort.
- **Refactoring**: Restructuring software to remove duplication, improve communication, simplify, or add flexibility without changing its behaviour; just-in-time redesign.
- **Pair programming**: A style of programming in which two programmers work side by side at one computer, continually collaborating on the same design, algorithm, code, or test.
- **Retrospective or reflection**: A post-iteration review of the effectiveness of the work performed, methods used, and estimates. The review supports team learning and estimation for future iterations.
- **Tacit knowledge**: Establishing and updating project knowledge in the participants' heads rather than in documents (explicit knowledge).
- **Test-driven development**: Developers and customers incrementally write module or method tests before and during coding. Supports and encourages very short iteration cycles. (Boehm B. T., 2005)

Obviously, Boehm's list contains many XP practices and therefore focused on more technical issues, while Highsmith's focus was more on organisational, team, customer and change oriented.

Therefore, they can be considered as complementary to each other as they cover different practice groups. As those were the starting-point practices for agile software development methods, the list has grown bigger and bigger by the time. With regard to the categorisation of agile practices, Boehm presented a systematic outline of agile practices by grouping them into 3 general areas:

- Communication (for example, metaphor and pair programming)
- Management (for example, planning game and fast cycle/frequent delivery)
- Technical (for example, simple design, refactoring, and test-driven design)

Similarly, Meyer has another classification of agile practices whereby he divided them under the categories of managerial, technical, and agile artefacts. As table 3 indicates, he has included so far the largest list of practices existing in academic literature.

Managerial	Technical	Agile artifacts
1.Sprint	12. Daily build and	19.Code
2. Daily meeting.	13. Continuous integration	20.Tests
3. Planning game, planning poker.	14. Test-driven development.	21. Use case, user story.
4. Retrospective.	15. Refactoring.	22 Burndown/burnup chart.
5. Collective code ownership.	16 Pair programming.	23 Story card/task card
6. Onsite customer	17. Simplest design	24 Story board/task board
7. Open space	18 Coding standards	25. Story points
8. Process Miniature		26. Velocity
9. Iteration planning		27. Definition of done
10. Review meeting		28. Working space
11. Scrum of Scrums		29. Product backlog, iteration backlog
		30. Impediment
		31. Waste, technical debt

Table 3: Classification of agile practices by Meyer (Meyer, 2014)

While Meyer presented the categorized list of agile practices, he failed to relate them to the principles he developed for the agile approach; it would give an added-value for his analysis and make it more insightful. Nevertheless, Meyer's analysis is one of the closest studies to the thesis research in terms of scope and addressed issues.

It is worth emphasising that, the list of agile practices is not limited to above mentioned classifications and practices. In fact, there is no evident list of agile practices that has been adopted as the industry standard: It is growing as agile methods are improving themselves and inventing new practices. For example, Jalali and Wohlin identified 25 agile practices, their diagram on frequency of agile practices in literature is interesting (Jalali, 2010). Moreover, VersionOne in its latest survey has used 25 agile techniques to find out which techniques are most used

(VersionOne, 2015). Finally, Agile Alliance has published exhaustive list of agile practices on its website. It is claimed to cover all agile methodologies and practices and consists of 60 agile practices. As displayed in figure 6, it is also presented as a mapping showing to which methodology or life cycle phase a certain practice belongs which consists of Extreme Programming, Scrum, Lean, Devops, Fundamentals, Team, Product Management, Design and Testing (AgileAlliance). For the thesis study, it has been chosen to use definitions and the list of practices of Agile Alliance, since it has been prepared by the authoritative organisation which unites most voices of the industry; therefore could also be considered as an industry standard. Furthermore, Agile Alliance continually conducts studies and updates its practice list, which is why it presents the most exhaustive and comprehensive list of practices. Therefore, the list of Agile Alliance could be considered as an industry standard.

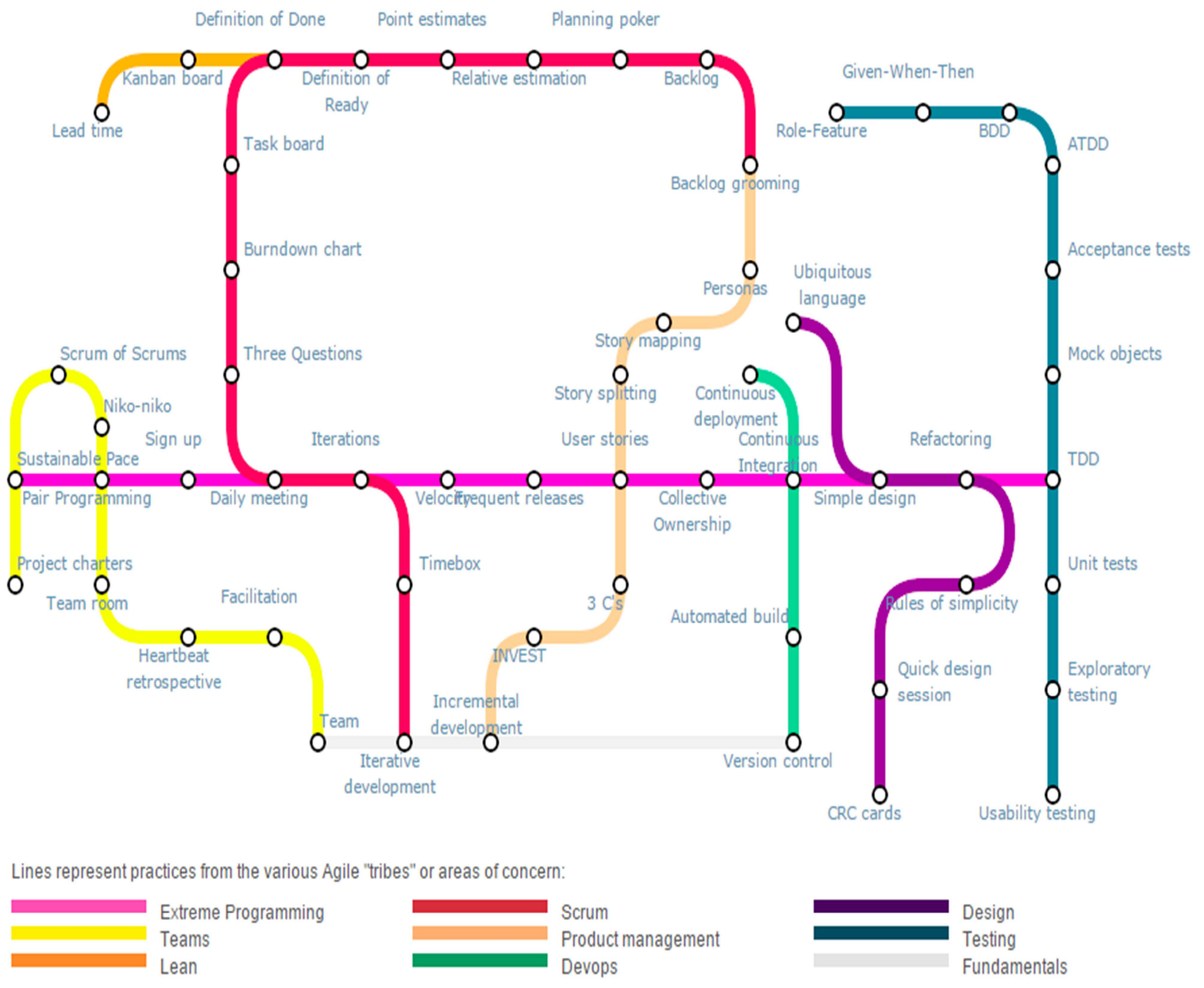


Figure 6: Agile Alliance mapping of practices according to the methods/phases they belong (AgileAlliance)

Of particular importance is Meyer's (2014) critical analysis of agile practices whereby he states that not all agile practices are good and benefits the development process. He categorises them as the brilliant, the good, the hype and the ugly/bad by pointing out their weaknesses and strengths. First list is **'bad'** ones which, according to Meyer, can damage the development process which include substitution of requirements with user stories, feature based development and ignorance of dependencies in difficult projects, rejection of traditional manager tasks and dependency tracking tools, embedded customer, coach as a separate role, test-driven development, deprecation of documentation. His **'hyped'** agile practice list contains very popular practices that have little value and benefit for the development process which include pair programming, open-space working arrangements, self-organizing teams, working at a sustainable pace, producing minimal functionality, planning game, planning poker, collective code ownership, and cross-functional teams. The **'good'** practice list of Meyer, on the other hand, includes refactoring, short daily meetings, team communication, removing impediments, and identification of sources of waste. The **'brilliant'** practices which he considers truly inspiring and effective contain short iterations, continuous integration, regression test suite, closed-window rule (freezing the iteration scope), time-boxing, product owner, delivering working software, velocity, task boards, and associating a test with every piece of functionality. (Meyer, 2014) No doubt, this criticism of Meyer provides an alternative and valuable insight on the usefulness and effectiveness of agile practices. Nevertheless, it is important to note that, Meyer has made this evaluation based on his own experience and thorough analysis of existing literature, therefore real empirical data supporting his arguments is missing which makes it vulnerable for validation threats. However, it presents an interesting level of evaluation which could be used as an analytical lens during the analysis phase of the study to see which category of practices the collected data supports.

2.4.6. Agile methods

According to Cockburn, a methodology is just the set of conventions a group of people agree to follow (Cockburn A. , 2007). By contrast, Livari&Maansaari suggest that a method may be interpreted not as a set of "rules," but as "an ideal in the sense that it is not expected to be followed literally." (Livari, 1998). From this point of view, agile methods are, in their essence, based on values and principles defined on the Agile Manifesto and composed of agile practices that have a certain synergy and have been created by experienced agile practitioners (Jalali, 2010). As it is mentioned earlier, agile is an umbrella term for different development methods which all share common values and principles of Agile Manifesto. A number of methods are included in this family, the most popular being *Scrum* (Schwaber, 2013)*, eXtreme Programming* (XP) (Beck, 2005), the *Dynamic Systems Development Method* (DSDM) (Stapleton, 1997)*, Crystal* (Cockburn A. , 2001), *Agile Modeling* (Ambler S. , 2002), *Feature Driven Design* (Coad, 1999), Adaptive Software Development (Highsmith, 2002), and *Lean software development* (LSD) (Poppendieck, 2001) (Conboy K. F., 2007). Kanban is also gaining popularity in recent years as an ASD method. Interestingly, these ASD methodologies were developed on three different continents: the Dynamic Systems Development Method in Europe; Feature-Driven Development in Australia; and Extreme Programming, Crystal, Adaptive Software Development, and Scrum in the US. (Dybå T. D., 2008) Larman&Basili identified Dynamic Systems Development Method (DSDM) as the first agile method, followed by extreme programming (XP). (Larman, 2003)

Perhaps more interesting is the fact that while some methodologies, like Scrum, focus more on project management aspect of development, using short, frequent team meetings to assess progress; others, like XP, focus more on development process itself, on technical Agile practices and is highly prescriptive and operational, by prescribing specific techniques such as pair programming (Barlow, 2011). On the other hand, Lean software development can be best described as a set of philosophical principles (Conboy K. , 2009). According to Smith, no methodology is better than the other, since each method focuses on specific values and there is no standard on how a method should implement its agility. Therefore, it all depends which works best in certain environment and within specific constraints. (Smith, 2009)

Alternatively, Conboy argues that agile methods are very disparate in character and differ greatly in terms of abstraction, goals, practices, and tools. Therefore it becomes difficult to compare methods and choose the 'right one' as a baseline method against others (Conboy K. , 2009). The reason might be the idea that each agile method carries only limited elements of agility which is also why they are often being combined in practice for an effective result. That is also the reason why the focus in this study lies on practices rather than methods, since the practice granularity provides a freedom of focusing on pure implementation rather than locking on certain method's boundaries. Moreover, Meyer similarly mentioned that many people also use some of the ideas/practices without embracing a complete method. Because according to him, every agile method is also an amazing mix of the best and the worst. That is why, instead of accepting single method entirely, agile teams make up its own cocktail of agile practices, rejecting the ones that do not fit. (Meyer, 2014)

It is not surprising that most popular combination of ASD methods is Scrum/XP mixture: while Scrum provides project management practices, XP practices are integrated for more development and technical purpose. On the other hand, some authors argue that, since each project is unique in character, they also require different approach. As Highsmith&Fowler put it, while variety and diversity of practices are necessary, when it comes to methodologies, there's no one-size-fits-all solution (Fowler, 2001). Furthermore, Highsmith&Cockburn (2001) advocate that agile methods offer generative rules – a minimum set of things you must do under all situations to generate appropriate practices for special situations. A team that follows generative rules depends on individuals and their creativity to find ways to solve problems as they arise. According to them, creativity, not voluminous written rules, is the only way to manage complex software development problems and diverse situations.[13] (Cockburn A. H., 2001)

For the purpose of this study, only the comparative views of agile methods will be presented without going deep into each of them. As a matter of fact, high level analysis of the methods will help to situate them better in agile discourse and have overall idea. From this point of view, the study of Abrahamsson et al is particularly interesting; they carried out a comparative study of 10 agile software development methods by trying to analyze them through the set of analytical lenses such as method's life-cycle coverage, project management support, type of practical guidance, fitness-for-use and empirical evidence (Abrahamsson P. W., 2003). They found out that, agile software development methods, without rationalization, cover only certain/different phases of the software development life-cycle and most of them do not offer adequate support for project management. As Figure 7 indicates, only DSDM and the Rational Unified Process were found to give full coverage to all phases of development, while Scrum mainly covers aspects related to project management (Abrahamsson P. W., 2003). Yet, many methods still attempt to strive for universal solutions (as opposed to situation appropriate) and the empirical evidence to support their claims is still limited.

[13] Most methodologies provide inclusive rules—all the things you could possibly do under all situations. Teams that follow inclusive rules depend on someone else to name in advance the practices and conditions for every situation. This obviously breaks down quickly (Cockburn A. H., 2001).

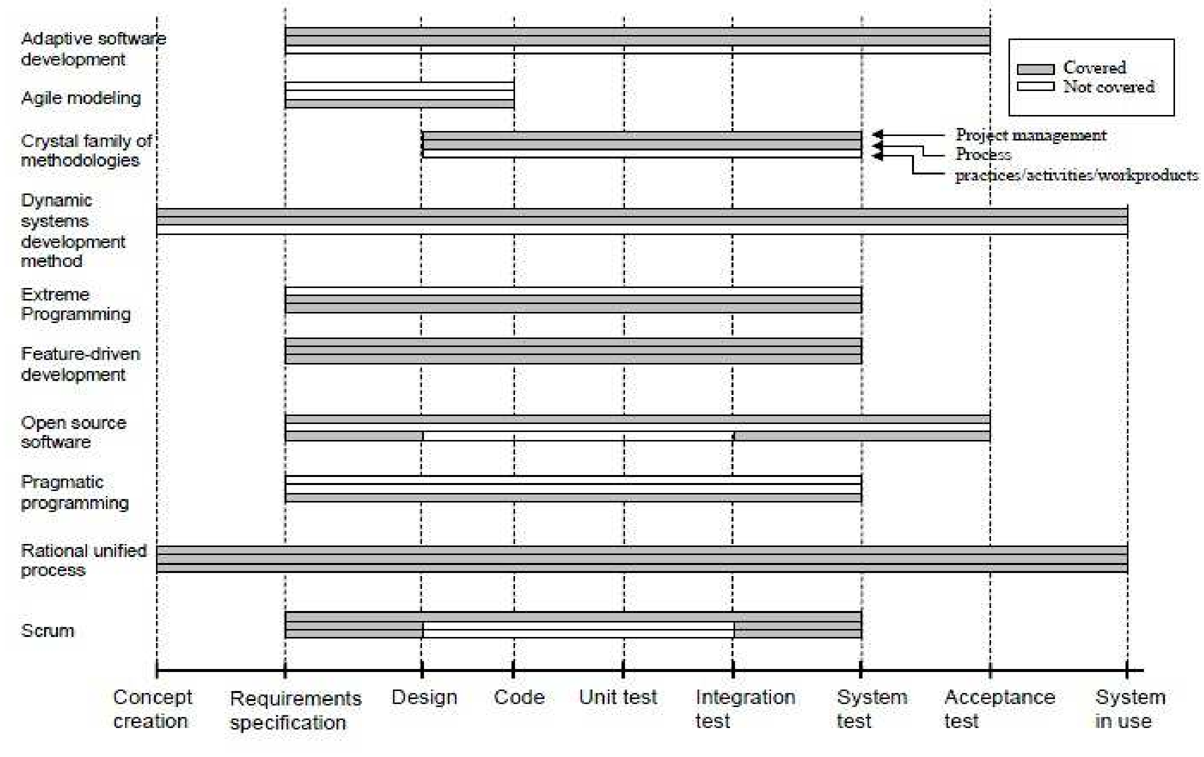


Figure 7: Comparing life-cycle, project management and concrete guidance support of agile methods (Abrahamsson P. W., 2003)

Of particular importance is the fact that, according to the survey of VersionOne, Scrum is currently the most adopted agile method with 56% use rate followed by Scrum/XP hybrid, Scrumban and Lean (VersionOne, 2015)]. Therefore, a comparative and descriptive summary of these 3 most popular agile methods could be helpful to get better understanding of them. Table 4 presents such a comparison which is developed by Kongyai and Edi (2011) containing of brief description, practices, process, roles and tools.

	XP	SCRUM	LEAN
Proposed/founded by	Kent Beck (1999)	Jeff Sutherland (1993) and Ken Schwaber (1996)	Mary and Tom Poppendieck (2003)
Reference Literature	*“Extreme Programming Explained”*	“*Agile Software Development in Scrum*”	*“Lean Software Development: An Agile Toolkit”*
Purpose	To address the specific needs of software	To manage software development project by	To put all development

	development performed by small teams facing vague and changing requirements.	organizing team and getting work done productively with higher quality.	efforts on value-adding activities from customers' viewpoint and to analyse and identify the waste in software process systematically and then remove it.
Key practices/ principles	1. The Planning Game 2. Small Releases 3. Metaphor 4. Simple Design 5. Testing 6. Refactoring 7. Pair Programming 8. Collective Ownership 9. Continuous Integration 10. 40-Hour Week 11. On-Site Customer 12. Coding Standards	1. Sprint Planning Meeting 2. Daily Scrum Meeting 3. Sprint 4. Sprint Review Meeting 5. Sprint Retrospective	1. Eliminate Waste 2. Build Quality In 3. Amplify learning 4. Defer Commitment 5. Deliver Fast 6. Respect People 7. Optimize the Whole
Approach	Iterative and Incremental	Iterative and Incremental	Continuous Improvement
Phases	1. Exploration 2. Planning 3. Iteration to Release 4. Productionizing 5. Maintenance	1.Sprint Planning 2.Sprint 3.Daily Scrum Meeting 4.Sprint Review Meeting 5.Sprint Retrospective	No phase involved
Main Roles	Programmer and Customer	Product Owner, Scrum Master and Team	No main role
Tools/Artifacts	User story card, Automated unit test, etc.	Product Backlog, Sprint Backlog, and Burn Down Chart	Kanban, Kaizen, VSM, etc.

Table 4: Summary comparison of main ASD approaches (Kongyai, 2011)

As Table 4 illustrates, these agile methods address different goals, use different practices, have distinct roles and phases and use various tools. More interestingly, Lean seems to have more philosophical principles instead of practices and it has no concrete roles. It is worth noting that, the inclusion of Lean into agile methodology umbrella has happened recently and it is getting more popular by time because of its universal applicable principles and simple tools like Kanban board. Put differently, its level of addressing issues are more high level and philosophical, while Scrum operates in project management level as a framework and XP, on the other hand, works on a technical level with its development-specific practices.

2.4.7. Criticism of agile software development

Despite its wide-spread prevalence, ASD has also been criticized by some practitioners and academics, mainly focusing on following five aspects:

1. Agile development is nothing new; such practices have been in place in software development since the 1960s
2. The lack of focus on architecture is bound to engender suboptimal design-decisions
3. There is little scientific support for many of the claims made by the agile community
4. The practices in XP are rarely applicable, and are rarely applied by the book
5. Agile development methods are suitable for small teams, but for larger projects, other processes are more appropriate. (Dybå T. D., 2008)

Most of these critics point specifically to: a lack of focus on planning and implementation, with too much focus on coding (Ambler S. , 2002); a lack of needed documentation that often results from decreased formal communication, and implementation failures in larger, more complex projects (Barlow, 2011). Likewise, Turk argues that, the emphasis on code can lead to corporate memory loss because there is little emphasis on producing good documentation and models to support software creation and evolution of large, complex systems. (Turk, 2002)

Equally interesting critical analysis of agile methods is pronounced by Turk; he has revealed the limitations of agile methods by linking those limitations to the self-stated assumptions that presumably exists within agile approach. As he goes further to argue, an appreciation of the (often unstated) assumptions underlying agile processes can lead to a better understanding of the applicability of agile processes to particular situations. Agile processes are less likely to be applicable in situations in which core assumptions do not hold (Turk, 2002). That means in order agile method to be successful, most of those assumptions must be fulfilled first, which in its turn generates limitations for agile methods. He elaborated the following limitations of agile methods, although he self agrees that not all of them are applicable for all agile methods:

1. Limited support for distributed development environments
2. Limited support for subcontracting
3. Limited support for development involving large teams
4. Limited support for building reusable artifacts

5. Limited support for developing safety-critical software
6. Limited support for developing large, complex software (Turk, 2002)

To sum up, while ASD methods are another attempt to find ‘a silver bullet’ solution for software project challenges, it does not necessarily claim to provide the same ‘silver bullet’ for all situations. Instead, it gives a choice of various methods and allows the adaptation as long as adherence to the basic values and principles of agile is maintained. In this regard, Medinilla (2012) describes the importance of the latter very blatantly. As he rightly states, “just trying to copy the practices and tools will not take the company into Agile. Only the constant effort to embrace the values and the principles will” (Medinilla, 2012). He further argues that, if the company is guided through the values and the principles in the Manifesto, it does not matter if it is doing Scrum, XP, Kanban, or Lean Software Development. As he perfectly puts it, ‘that it is a matter of stop doing agile and instead begin being agile’. (Medinilla, 2012) According to him, agile companies must understand values and principles first and then try to find the set of processes, roles, practices, artefacts, and tools that help them live by those values and principles and not the other way round. (Medinilla, 2012)

3. EMPIRICAL RESEARCH

The objective of this research study is to find out the implementation degree of agile practices, the perceived importance and real presence of agile principles in agile software development process. While choosing the research methodology, its capacity to enable the researcher to reach the research objectives should be evaluated. To be able to answer the research questions, 2 kinds of data will be used for analysis and discussion: secondary and primary.

3.1. Secondary Research

Secondary research covers the study of all related literature and also data from related surveys. It supplies 2 kinds of data: first, theoretical base and background for analyzing the primary data; second, secondary empirical data from related surveys on RQ1 and RQ2, for backing up its own primary data in analysis phase. Theoretical part is thoroughly discussed in previous chapter, and therefore will not be covered here.

3.1.1. Related surveys

The secondary empirical data is obtained from different surveys that is conducted by various organizations and experts and covers quite high number of respondents worldwide. Those are VersionOne survey, AmbySoft survey, and several substantial studies of Laurier Williams, which includes worldwide surveys.

3.1.1.1. VersionOne Survey

VersionOne conducts each year survey since 2006 which has provided an aggregate report on the status of organizations currently implementing or practicing agile methods. The 9th annual State of Agile survey was conducted between July and October, 2014. Sponsored by VersionOne, the survey invited individuals from a broad range of industries in the global software development community. A total of 3,925 completed responses were collected, analyzed and prepared into a summary report by Analysis.Net Research, an independent survey consultancy. (VersionOne, 2015)

It includes data on Company Experience and Adoption, Benefits of Agile, Agile Methods and Practices, Agile Success and Metrics, Scaling Agile, Project Management Tools. For the purpose of this study only the data related to agile practices will be used to back up own survey results. VersionOne describes practices as techniques and according to its survey results the most widely practiced agile technique is the daily stand-up (80%), followed closely by the use of short iterations (79%) and prioritized backlogs (79%). About two-thirds of respondents said they conduct Iteration planning and retrospectives, while less popular techniques included agile games (13%) and Behaviour-Driven Development (BDD) (9%). The full list of practices and corresponding percentages are given below: (VersionOne, 2015)

80% Daily stand-up
79% Short iterations
79% Prioritized backlogs
71% Iteration planning
69% Retrospectives
65% Release planning
65% Unit testing
56% Team-based estimation
53% Iteration reviews
53% Task board
50% Continuous integration
48% Dedicated product owner
46% Single team (integrated dev & testing)
43% Coding standards
38% Open work area
36% Refactoring
34% Test-Driven Development (TDD)
31% Kanban
29% Story mapping
27% Collective code ownership
24% Automated acceptance testing
24% Continuous deployment
21% Pair programming
13% Agile games
9% Behaviour-Driven Development (BDD)

It is worth noting that, VersionOne survey has the most recent data on practice usage, however VersionOne limits it only to 25 practices and it has not surveyed anything on importance of agile principles.

3.1.1.2. AmbySoft Survey

This survey is conducted by Scott Ambler, the founder of Agile Modeling method, in July 2008. It is about the usage of agile practices and principles. 337 respondents have answered the survey. He has grouped agile practices in 4 categories and asked accordingly how common those practices are practiced in the respondent’s projects. Results of practice usage are formulated in -5 to 5 scales rating. The results are reported in Table 5.

Project Management practices	Development Practices	Quality practices
• Iteration planning (3.54) • Daily Scrum Meeting (3.29) • Prioritized worklist (3.08) • High-level release planning (2.19) • Retrospectives (1.84) • One Product Owner (1.55) • Burndown chart (1.51) • Potentially Shippable Software (1.51) • Status Reports (1.15) • Story Board with Task Breakdowns (0.83)	• Coding Standards (2.30) • Collective Code Ownership (1.97) • Continuous integration (1.94) • Database standards (1.86) • UI standards (1.65) • Pair programming (-1.34)	• Code Refactoring (1.79) • UI Testing (1.54) • Automated Developer Testing (1.08) • TDD (-0.08) • UI Refactoring (-0.22) • Database refactoring (-0.31) • Automated Acceptance Testing (-0.87) • Database regression testing (-1.03) • Executable Specs (-1.43)

Table 5: Practicing rates of agile practices (Ambler S. V., 2008)

He also constructed statements related each agile principles and asked if the respondent agrees with that statement. Most of the statements received highly positive answers, most people saying they agree or strongly agree with the principle statements which show that practitioners still support principles of agile manifesto. Overall, it proves to be very valuable survey, which covers both agile principles and practices. However, it doesn't cover all agile practices, doesn't measure the importance of agile principles and also don't give any correlation between agile principles and practices.

3.1.1.3. Laurie Williams' studies

Laurie Williams, associate professor of computer science at North Carolina State University, conducted a survey to obtain input from the international practitioner community on the original principles and their associated software development practices. 335 people from around the work responded to the survey. Respondents were asked to say how important each principle was to agile teams in 2010 on a scale from 1-5 with a rating of 1 indicating not important and a 5 being very important. According to the survey results, as Williams states, the community still stands strongly behind all the twelve principles. In round numbers, 11 of the principle had at least 80% of the respondents giving the principle a rating of 4 or 5. The Agile Principle that had the least support was "The best architectures, requirements, and designs emerge from self-organizing teams" with only 59% of respondents giving this principles a 4 or 5. (Hastie, 2010) Among the findings were that the following are the most important principles based on the number of respondents rating their importance as “Very High”:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Working software is the primary measure of progress.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
5. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. (Cohn, 2010)

The survey also asked which agile project management practices were essential for a team to be considered agile. The top five were:

1. Short iterations
2. Continuous integration
3. “Done” criteria
4. Automated tests are run with each build
5. Automated unit testing. (Cohn, 2010)

Another survey, which is discussed in Williams’ article on “Agile software development in Practice”, is the industry-wide conducted survey based upon 2,229 completed CA surveys taken between March 26, 2011 and October 12, 2012. The Comparative Agility™ (CA) assessment tool developed by Williams self, was created to aid organizations in understanding how their agile adoption compares with others’ agile adoption. At the highest level, the CA approach assesses agility across eight dimensions: Teamwork; Requirements; Planning; Technical Practices; Quality; Culture; Knowledge Creating; and Outcomes. The survey respondent is presented with 65 statements. Each statement is an agile practice for which the respondent indicates the truth of the statement relative to their team or organization. (Williams L. D., 2014) She has already done the same survey during 2007-2010 and got very interesting results.

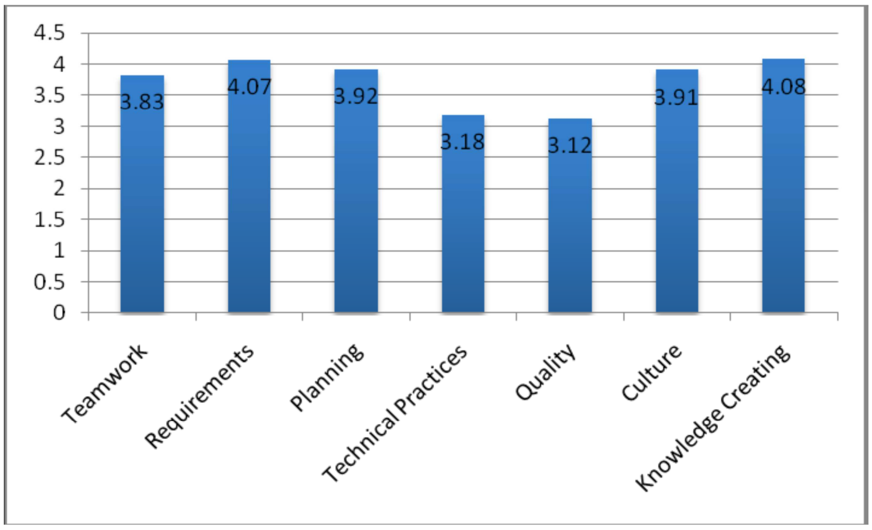


Figure 8: Industry average responses[14] **(Williams L. R., 2010)**

Figure 8 displays that industry trends indicate the highest adoption of agile practices occur in the areas of embracing emergent requirements and creating knowledge throughout the iteration and release. The lowest industry adoption occurs relative to utilizing technical practices and focusing on quality throughout all iterations. (Williams L. R., 2010)

Additionally, she mapped the CA statements to one of the twelve agile principles defined in the Agile Manifesto. Although statements may address more than one principle, each was assigned to the one principle that the statement most influenced. And afterwards, she made the analysis of how agile principles' are practiced, based on statements and mapping. She found out that all but three principles (emergence, excellence, and simplicity) report at least 50% of teams are practicing the agile principle. The agile principles with the largest positive results are Individuals, Business, and Face-to-face. (Williams L. D., 2014) Moreover, she made the list of most popular and least popular 5 statements/practices.

Surveys and studies/analysis of Williams are highly related to this study, since her first survey measures the importance of agile principles and practices for agility. Second/third survey is more about to assess agility through the survey tool and is more practice oriented. However, it names practices differently and don't measure the importance degree of principles. On the other hand, she has made correlation of agile principles and practices in her studies. On her last research paper, she mapped agile principles with practices and based on the assessment survey, she made

[14] The number presented for each dimension is the mean score of the responses on the set of statements related to that dimension whereby for each question a response of True=5; More true than false=4; Neither true nor false=3; More false than true=2; and False=1.

conclusions on principle practicing degree of respondents. (Williams L. D., 2014) This means she analyzed presence of agile principles in agile process. However, she didn't couple her importance survey results with presence results. Her survey respondents were mainly teams that newly adopted agile and had few experience with agile practices. This brings also limitations to its generalization.

The results of these surveys and analyses will be used to back up and check the results of this study to make it more reliable and valid. However, it should be kept in mind that they are not designed specifically to answer the research questions of this study and therefore contains some variations and therefore will be used with reservations.

3.2. Primary research: selecting the relevant research method

In order to investigate perceptions and realities on importance of agile principles and practices, following methodology approach seems to be suitable: agile experts are consulted for mapping agile principles and practices in a manner that they consider relevant to each other. Moreover, a survey on implementation degree of practices in real ASD process and perception of importance of agile principles is conducted separately. That enables a two dimensional vision to analyze data and be able to make correlations more accurately. While mapping provides an idea about which practice is based on which principle, survey, on the other hand, delivers data to analyze the results of mapping, which means if it matches with reality/perceptions of practitioners.

Taking into account all of this, quantitative strategy has been chosen for conducting the primary empirical research. According to Bryman, quantitative research can be construed as a research strategy that emphasizes quantification in the collection and analysis of data and that: - entails a deductive approach to the relationship between theory and research, in which the accent is placed on the testing of theories; - embodies a view of social reality as an external, objective reality (Bryman, 2004). While quantitative research is based on numerical data analyzed statistically, qualitative research uses non-numerical data, emphasizes words, and stresses the subjectivity of research process (Creswell, 2003). It is important to distinguish the rationale behind the choice of research strategy, because they support different causes. Whereas quantitative methods are best for looking at cause and effect, qualitative methods are more suited to looking at the meaning of particular events or circumstances (Creswell, 2003). Bryman also believes that choices of research strategy, design, or method have to be dovetailed with the specific research questions being investigated. According to him, if we are interested in teasing out the relative importance of a number of different causes of social phenomenon, it is quite likely that a quantitative strategy will fit our needs, because the assessment of cause is one of its keynotes (Bryman, 2004). In the case of this study, it is aimed to find out the implementation degree of practices and importance degree of principles for practitioners, and actual presence of agile principles in agile development which means quantification will be needed to measure these variables. Therefore, quantitative strategy suits best for this research study.

When using quantitative strategy, it should be remembered that, the fact that the data have to be quantitative does not mean that they have to be naturally available in quantitative form. This kind of data can be still collected in a quantitative way. It can be done by designing measurement instruments aimed specifically at converting phenomena that don't naturally exist in quantitative form into quantitative data, which can be analyzed statistically. If data are not naturally available as numbers, one can try to turn non-quantitative data (like attitudes or opinions) into quantitative data by measuring them numerically (for example, by using a questionnaire rating scale). (Muijs, 2004)

Bryman distinguishes research design from research method. According to him, research design provides a framework, structure for the collection and analysis of data, while research method is simply a technique for collecting data. Research methods are associated with different kinds of research designs. He further elaborates five different kinds of research designs: experimental design, cross-sectional or survey design, longitudinal design, case study design and comparative design (Bryman, 2004). For this study, cross sectional or survey design will be used to collect empirical data. By Bryman's definition, a cross sectional design entails the collection of data on more than one case and at a single point in time in order to collect a body of quantitative/quantifiable data in connection with two or more variables, which are then examined to detect patterns of association. Survey research is a cross-sectional design whereby data are collected predominantly by questionnaire or by structured interviews. (Bryman, 2004)

3.2.1. Survey as a main research method

Surveys are appropriate when we want to learn about self-reported beliefs or behaviours. Most surveys ask many questions at once, thereby measuring many variables, with the intent of generalizing from a sample to a population. Surveys can be used for exploratory, descriptive, or explanatory research and employ different methods such as web surveys, surveys by post, self-completed questionnaires, and structured interviews. (Neuman, 2011)

Survey is chosen as a main research method for RQ1 and RQ2. Obviously, survey is considered to suit best for this study's descriptive objectives which aims to explore a certain situation and describe the important factors associated with that situation (Neuman, 2011). A survey was conducted to figure out what agile practitioners think about the importance of agile principles. It also investigates the usage/implementation degree of each practice by practitioners. According to Robson, survey is the “collection of standardized information from a specific population, or some sample from one, usually, but not necessarily by means of a questionnaire or interview” (Robson, 2002). A web/online survey was preferred as the data collection method because it is considered to be the most efficient way of gathering large data sets in short span of time. Moreover, online questionnaires are able to reach more potential respondents, given that the questionnaires can be distributed through many channels (e.g. e-mail, forums, and communities); and are very fast and inexpensive; they allow flexible design and can use visual images and even audio or video. It has static and interactive types. (Neuman, 2011) In addition, the relative low cost of conducting a web survey essentially puts the tool in the hands of almost every person with access to the Internet, potentially fully democratizing the survey-taking process (Couper, 2000). On the other hand, disadvantages of web surveys are coverage, privacy and verification, and design issues. Being cheap and easy to implement can be also disadvantage, since it affects quality of the survey. (Neuman, 2011)

3.2.2. Mapping Study

Another research method that will be used to help answer the research questions is mapping study. In fact, mapping is chosen for RQ3. It is hoped to enable the author to find sound correspondence match conclusions based on this mapping exercise. It aims to generate a theoretical basis for our analysis and answer for our research question 3. It is relatively new method of research and contains in itself very high level correlation capabilities. It is chosen since it is simple to follow and enables to gain the knowledge from experts in a more straightforward way. Although mapping itself is a qualitative form of gathering data, the analysis of collected data will be done quantitatively, and will be quantified in order to be able to make comparisons and further analysis in combination with other data. Based on an analysis of secondary data, mapping results and the survey results, conclusions regarding agile principles’ presence and importance will be derived. Thus for answering RQ4, results of all above mentioned methods will be combined and analysed to get better understanding.

If summed up, the table of mapping research methods and corresponding research questions, that it aims to answer, could be presented as in Table 6. The online survey is used as the methodological triangulation to answer RQ 1 and RQ 2, which is also backed up and possibly validated by secondary data from other relevant surveys. Triangulation is necessary to increase the accuracy of empirical research. By using the triangulation, researcher can take different angles towards the studied object and thus get a broader picture (Neuman, 2011).

Research questions	Used Research method
RQ1	Online Survey, secondary data
RQ2	Online Survey, secondary data
RQ3	Mapping
RQ4	Analysis of online survey results and mapping data together

Table 6: Linking research methods to research questions

3.3. Evaluating the chosen methods: Limitations and Concerns

According to Bryman, three of the most prominent criteria for the evaluation of social research are: reliability, replication, and validity. Reliability is concerned with the question of whether the results of study are repeatable. In order replication to take place, a study must be capable of replication – it must be replicable (Bryman, 2004). Surveys, especially web-based ones, offer little control over the interview situation, why there might be low reliability in the survey results (Couper, 2000). The reliability of the survey was strengthened by the fact that the questions were not of a particular sensitive nature and no reason is seen for respondents to give incorrect answers, especially when considering that the survey was taken anonymously. Most important criterion of research is validity. Validity is concerned with the integrity of the conclusions that are generated from piece of research. When it is said that indicator is valid, it is valid for a particular purpose and definition. The same indicator may be less valid or invalid for other purposes. Reliability refers to a measure's dependability; validity refers to its truthfulness or the fit between construct and data. (Neuman, 2011) The validity is a concern of every researcher as it makes the research valuable in different terms. Any research based on empirical findings does possess concerns and so is our research. The following concerns will be discussed for the chosen research methods:

3.3.1. External validity

External validity mainly deals with the ability to generalize the research away from the scope of the study. In quantitative research the researcher is usually concerned to be able to say that her findings can be generalised beyond the confines of the particular context in which the research was conducted (Bryman, 2004). Selecting wrong set of sample population to represent the whole target population is a serious threat. In order to address this threat, certain groups have been considered and personal invitations to respondents and researchers who are related to our research study have been sent. Moreover, the responses obtained from the survey were from different systems types and also with different team sizes. So, in terms of diversity it can be generalised, however, in terms of numbers of responses, some doubts can be emerged as it can be the case that it doesn't reach the wanted numbers. In such case, to neutralise this low number effect, the other survey results will be used to reflect the results of this survey.

3.3.1.1. Low response rate of survey

In surveys there is always a relatively high risk of non-respondents and this can threaten the validity of the survey results (Bryman, 2004). The low response rate is a possible source of bias as they are only from representative of those who replied. To ensure a sufficient number of responses, some techniques were used to increase the response rates. For example, easy-to-understand and as clear as possible survey questions were created and also reminders to the invited participants were sent out. Furthermore, an incentive of providing the study results to respondents was offered to get better response rate. As mentioned above, in case of low response, in order to strengthen external validity of the study results, the secondary data from other relevant surveys will be used as a back-up during analysis and discussions to examine if those data sets match with each other or if they have major differences.

3.3.1.2. Selection bias

Selection bias means that particular individuals or groups have more chance to participate in the survey than others, resulting in biased samples. As this survey target population is from Agile practitioners, numerous Agile group communities were invited, which were randomly searched from the Internet. By using this technique, every Agile group would have equal chance to participate in the survey. In other words, the individual within the groups would have equal chance to be selected (Couper, 2000).

3.3.1.3. Unclear survey questions/instructions

Another threat to the validity of surveys is that the questions may not capture the real perceptions of the survey respondents. Since there is no direct involvement of the researchers when the respondents taking the survey, unclear survey questions could not be easily clarified and the respondents might not follow the instructions (Rea, 2005). To minimize this threat, survey questions were based on research questions. Furthermore, clear and brief description of each agile practice is also given so that everybody understands the same definition under certain terminology and answers accordingly.

3.4. Research design process – data collection

3.4.1. Survey set up/design

For any survey, it is important to identify the objectives of the survey on which the questionnaire has to be developed and to select the target population on which the sample has to be collected. Failing to do these may lead to very less as well as irrelevant data. In this survey, main objectives are to target the population who possess sufficient knowledge and experience in the field of "Agile software development" implementation, and to ask them questions that reflect the research objectives. People from all over the world can participate in the survey, as this is web based survey. The survey questions were designed carefully to ensure they can help to answer the research questions. The web-based survey contains 20 questions which is divided into 3 main parts: demographics, agile practice implementation, and agile principles' importance. The full design of the survey is shown in Appendix A.

The demographics of the respondents are addressed by question one to seven. It covers the following information: Position/role, company profile, size, project size, team size and so on. Second section contains question on the implementation degree of 60 agile practices. For every agile practice, brief description is given in order to make sure that everybody understands the same definition when answering the questions. For this end, definitions of AgileAlliance are used to maintain consistency and follow the standards. Responses were made using five-point Likert scale ranging from "not applicable" to "fully implemented". Likert scale is one of the most common techniques for conducting investigation of attitude which is a prominent area in survey research. Named after Rensis Likert, who developed the method, the Likert scale is a set of attitudes relating to a particular area. The goal of Likert scale is to measure the intensity of feelings about the area in question (Bryman, 2004). Third section is about how important are agile principles for practitioners; it is an attitude measurement question and therefore also contains Likert scale. Although this survey is anonymous, there has been provided an option to enter an email address if the respondent is interested to receive the survey results. After the survey design was reviewed, finalized and updated on the online survey tool, the survey was posted online. The invitation emails were sent to agile practitioners from many different countries. Several online discussion groups e.g. LinkedIn, Yahoogroups, Googlegroups, and Google+ Communities were identified that focus on agile software development and survey link with accompanying message was posted inviting the group members who had experience using Agile approaches. The survey invitation can be found in Appendix C.

3.4.2. Mapping design

The mapping is used to identify the possible correspondence/match between agile principles and practices. It is designed in Excel which contains 12 agile principles and 60 agile practices as a matrix. For the ease of use, this matrix is divided into 6 smaller ones in separate sheets but still in the same document. For each practice, the same brief description like in the survey is given to make it consistent and understandable. Afterwards, it was sent as an email attachment with accompanying explanation letter to the different agile experts and academicians who have extensive knowledge and expertise on agile software development. The mapping design and invitation letter can be found in Appendix B and D. They were asked to map each agile practice with relevant principles by determining which agile practice possibly supports/based on which principles.

4. EMPIRICAL FINDINGS

4.1. Survey results and analysis

Generally, descriptive statistics is used to analyse the collected data from survey and mapping. Descriptive statistics is used to describe the basic features of the data in a study; it helps to simplify large amounts of data in a sensible way and provide a powerful summary that may enable comparisons across people or other units (Descriptive Statistics). Furthermore, the most used descriptive statistics measures are frequency distribution, mean score, mode and variance. It is vitally important to note that data is not tested for significances; therefore interpretations are also made with reservations.

In this section, findings from survey demographics, implementation of agile practices, and importance of agile principles will be presented and analysed. In total, about 200 survey invitations were sent out to agile practitioners by email. In addition, survey link was posted in approximately 30 agile discussion platforms such as LinkedIn groups, Yahoo Groups, Google Groups, and Google+ Communities. Despite so many postings and invitations, only 72 responses were received, from which 68 have been considered to be valid after filtering. Most certainly, the reason for this low ratio between response rate and number of invitations would be the time factor for completing the survey. This low rate of response doesn't give the opportunity to make accurate generalizations. Nevertheless, the overall responses obtained from the survey are sufficient to answer our research questions and make it possible to draw valuable conclusions. Moreover, this data will be backed up by the secondary data from other similar surveys conducted by professionals mentioned above. It is important for any survey to make a trade-off between superficial data and a short survey with many responses, and very rich data with fewer answers. In this case, it has been chosen to cover as many as possible agile practices and therefore ended up with a long list of practices. It should be noted that not all questions were answered by all 68 respondents. Some respondents have left most questions blank and unanswered and therefore have been filtered from the analysis. Other group of respondents answered only demographic questions and then left the survey; other ones have not answered the last question on principles. Therefore, the number of respondents varies per section and sometimes per question, because some respondents chose to skip some of the questions.

4.1.1. Respondents' Demographics

4.1.1.1. Respondents' role and experience

Survey respondents identified themselves in various roles and positions: 35% of them were different level managers, 28% classified their role as Agile Coach/Scrum Master, 14% as developer/IT specialist, and 12% as different kind of consultants within IT domain. According to survey results, the number of respondents working in multinationals was slightly higher than those working in local companies (56% - multinational companies, 44% - local companies). Furthermore, for 55% of these companies, software development is a main business activity, while for 45% it is an internal activity. The experience of respondents with agile development fluctuates as well: Figure 9 indicates that survey respondents have an average of 7.2 (mode = 5) years of agile experience. 22% of them have 1-2 years of experience with agile, while 25% has 3-4 years of experience. 15% of them have 8 years of agile experience. Another remarkable group of respondents is those having more than 10 years of experience consisting 17% of respondents.

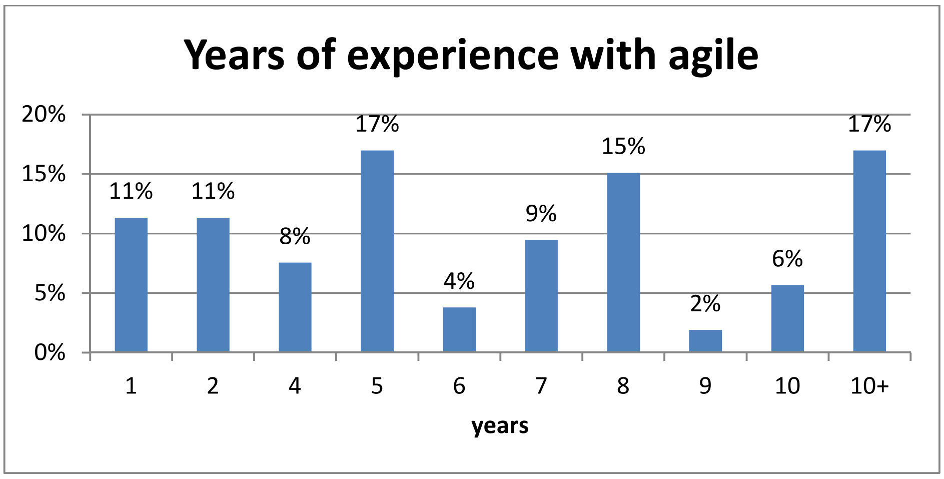


Figure 9: Respondents' experience with agile

4.1.1.2. Company and team size

Company size of participated respondents is dominated by large companies. Overall, the respondents' answers fell into six major categories based on company size: it is interesting to note that 40% of concerned companies are large organisations which have more than 1000 workers. Only 19% of them are small sized-companies having 1-10 workers. And 21% has 11-100 being the middle size category; 16% has 201-1000 workers which also can be considered as the large organisations. Table 7 displays the categorization according to company size.

company size	number of respondents	percentage
1--10	11	19%
11--55	9	16%
56-100	3	5%
101-200	2	4%
201-1000	9	16%
1000+	23	40%
Total	57	100%

Table 7: Company size of participated respondents

It is also important to know the size of the respondents' team to draw valid conclusions. As for team size, more than half of respondents (55%) works with small teams (1-10), 23% with medium teams having 11-100 team members, and surprisingly 22% with large teams (more than 100 members). Even 8% of them work with teams which have more than 1000 team members. Figure 10 presents these results in a more visible way.

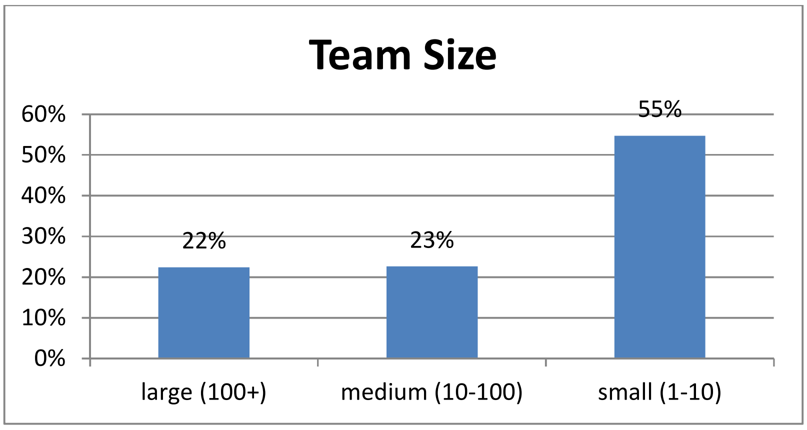


Figure 10: Team size of participated respondents

The last data on demographics cover the average length of iterations. According to the survey results, more than half of the respondents have 2 weeks long iteration in their software project (59%). Next most popular iteration length is 4 weeks with 13% of respondent pool. There are even some projects that have less than 1 week iteration period (4%). 1 and 3 weeks long iterations have each 7% respondent vote. And last, respondents that work with iterations longer than 10 weeks consist 6% of all answers.

4.1.2. Agile Practice implementation

Survey respondents evaluated their implementation degree of 60 agile practices presented in survey according to 5-point Likert scale with following options: not implemented (<20%), partially implemented (20-50%), mostly implemented (51-80%), fully implemented (>80%), and not applicable. Not applicable option has been included for the cases where the practice is not relevant in certain situations (can be due to used method or circumstances). In order to make statistical analysis of this scale, it has been coded into numbers, as following: "not implemented" = 1, "partially implemented" = 2, "mostly implemented" = 3, "fully implemented" = 4, and "not applicable" = 5.

Accordingly, frequency distribution is calculated in percentages to show the degree of implementation for each practice. As Table 8 indicates, percentages of implementation for some practices are substantially higher (Backlogs, Daily meeting), while for others it becomes remarkably lower CRC cards, ATTD).

List of practices	not implemented	partially implemented	mostly implemented	fully implemented	not applicable	total responses	mean	mode	variance
Acceptance tests/functional test	14%	18%	23%	41%	5%	44	2.95	4	1.35
ATDD (acceptance test driven development)	50%	14%	14%	9%	14%	44	1.79	1	2.23
Automated build	11%	23%	20%	41%	5%	44	2.95	4	1.30
Backlogs (Product and Sprint)	2%	5%	14%	77%	2%	44	3.70	4	0.48
Backlog grooming	5%	16%	20%	57%	2%	44	3.33	4	0.89
Behaviour-driven development (BDD)	57%	14%	7%	7%	16%	44	1.57	1	2.38
Burndown Chart	20%	14%	14%	45%	7%	44	2.90	4	1.72
Collective code ownership	18%	16%	30%	30%	7%	44	2.76	4	1.48
Continuous integration	16%	16%	23%	43%	2%	44	2.95	4	1.35
CRC Cards (for Class, Responsibilities, Collaborators)	64%	9%	2%	5%	20%	44	1.34	1	2.74

Continuous deployment	34%	20%	25%	14%	7%	44	2.20	1	1.64
Daily meeting (daily scrum or stand up)	7%	14%	2%	75%	2%	44	3.49	4	1.00
Definition of done	9%	30%	14%	45%	2%	44	2.98	4	1.23
Definition of ready	32%	20%	11%	30%	7%	44	2.41	1	1.92
Estimation	11%	11%	23%	50%	5%	44	3.17	4	1.22
Exploratory testing	27%	20%	15%	27%	12%	41	2.47	1	2.03
Facilitation	17%	20%	20%	41%	2%	41	2.88	4	1.42
Frequent releases	7%	15%	22%	54%	2%	41	3.25	4	1.01
Given-when-then	34%	15%	22%	12%	17%	41	2.15	1	2.24
Heartbeat retrospective (sprint/iteration retrospective)	5%	15%	22%	56%	2%	41	3.33	4	0.89
Information radiators	7%	41%	15%	27%	10%	41	2.68	2	1.39
Integration	7%	29%	20%	37%	7%	41	2.92	4	1.27
Invest acronym INVEST	22%	24%	27%	10%	17%	41	2.29	3	1.89
Iteration/sprint	0%	2%	15%	76%	7%	41	3.79	4	0.31
Iterative development	0%	10%	29%	59%	2%	41	3.50	4	0.50
Incremental development	2%	15%	17%	63%	2%	41	3.45	4	0.76
Kanban board	10%	24%	17%	32%	17%	41	2.85	4	1.63
Milestone retrospective	32%	20%	15%	20%	15%	41	2.26	1	2.18
Mock objects	27%	17%	24%	22%	10%	41	2.46	1	1.81
Niko Niko calendar (mood board)	66%	10%	5%	2%	17%	41	1.32	1	2.40
Lead/cycle time	10%	22%	32%	20%	17%	41	2.74	3	1.51
Pair programming	27%	46%	12%	7%	7%	41	2.00	2	1.33
Personas	40%	28%	8%	15%	10%	40	1.97	1	1.95
Points (estimate in)	20%	7%	17%	46%	10%	41	3.00	4	1.71
Planning poker	27%	17%	22%	22%	12%	41	2.44	1	1.94
Project chartering	32%	22%	15%	15%	17%	41	2.15	1	2.24
Quick design	20%	27%	22%	22%	10%	41	2.51	2	1.64

session									
Refactoring	7%	32%	20%	39%	2%	41	2.93	4	1.12
Relative estimation	27%	22%	15%	27%	10%	41	2.46	1	1.91
Role-feature reason	41%	7%	10%	29%	12%	41	2.31	1	2.44
Rules of simplicity	46%	10%	5%	17%	22%	41	1.91	1	2.90
Scrum of scrums	29%	12%	12%	12%	34%	41	2.11	5	2.84
Simple design	35%	25%	15%	13%	13%	40	2.06	1	1.99
Sign up for tasks	15%	29%	17%	34%	5%	41	2.74	4	1.43
Story splitting	12%	27%	20%	39%	2%	41	2.88	4	1.27
Story mapping	41%	22%	10%	15%	12%	41	1.97	1	2.13
Sustainable pace	15%	24%	15%	39%	7%	41	2.84	4	1.55
Task board	2%	12%	15%	66%	5%	41	3.51	4	0.75
Test-driven development (TDD)	41%	22%	12%	17%	7%	41	2.05	1	1.85
Team (whole)	12%	12%	20%	51%	5%	41	3.15	4	1.29
Team room	22%	15%	15%	46%	2%	41	2.88	4	1.62
Three C's	34%	20%	7%	22%	17%	41	2.21	1	2.42
Time-box	12%	5%	27%	51%	5%	41	3.23	4	1.17
User stories	7%	15%	10%	66%	2%	41	3.38	4	1.05
Three questions	15%	15%	10%	56%	5%	41	3.13	4	1.48
Usability testing	24%	22%	15%	29%	10%	41	2.54	4	1.88
Ubiquitous language	25%	20%	15%	23%	18%	40	2.42	1	2.16
Unit testing	8%	20%	20%	53%	0%	40	3.18	4	1.02
Velocity	20%	15%	10%	48%	8%	40	2.92	4	1.76
Version control	5%	10%	13%	70%	3%	40	3.51	4	0.82

Table 8: Frequency distribution of implementation degree of agile practices

In order to analyse the practice implementation further, the statistical mean score is calculated to find the average for each practice. The mean or average is probably the most commonly used method of describing central tendency and it is highly suitable to get overall high level picture of implementation degree of all practices. Firstly, count the total of weighted values by multiplying the respondent number with the column weight (column weight for "not implemented" = 1, "partially implemented" = 2, and so on). Secondly, count the total number of respondents. Finally, calculate the rating average by dividing the total weighted value by the total number of respondents. It is important to note that the option "not applicable -5" is not included while calculating the mean, because it doesn't contribute to the implementation degree of practices. For example, the calculation for acceptance test mean score is ((1*6)+(2*8)+(3*10)+(4*18)) / (6+8+10+18) = 124 / 42 = 2.95. An average rate of 2.95 means that acceptance test overall implementation falls to almost "mostly implemented" rating. Table 7 provides additional information on mean, mode scores and variances. Various analyses can be made based on mean, because it provides us with a powerful average implementation degree of each practice. The mode, on the other hand, describes the most frequently occurring value among the given options. From the mode data, it can easily be observed which implementation degree is chosen the most for each practice. However, it should be kept in mind that, as the data has not been tested for statistical significance, the interpretations should be made also with care and limit. Variance is used to compensate it in certain level to detect the controversial data.

4.1.3. Agile Principles importance

Third part of the survey was about the learning the attitude of respondents on importance of agile principles. 12 agile principles were presented to respondents and they were asked to indicate how important they consider each principle according to 6-point Likert scale. It is also coded in order to be able to make statistical calculations/measurements and analysis as following: "unimportant" = 1, "partly unimportant" (low importance) = 2, "neither unimportant nor important" (neutral) = 3, "partly important" = 4, "important" = 5 and "very important" = 6.

The frequency distribution per principle is reported in Table 9 in the form of percentages. For the ease of use, every principle is mentioned with keywords that characterise it most.

Principles	unimportant	partly unimportant	neither important nor unimportant	partly important	important	very important	total responses
1. Customer satisfaction/outcome	0%	3%	3%	21%	23%	51%	39
2. Welcome change	3%	5%	8%	18%	31%	36%	39
3. Deliver frequently with shorter timescale	0%	3%	3%	21%	33%	41%	39
4. Business and developers' daily collaboration	3%	5%	15%	18%	28%	31%	39
5. Support and trust motivated individuals	5%	0%	8%	21%	23%	44%	39
6. Face-to-face	3%	8%	8%	5%	33%	44%	39

communication							
7. Working software is progress	3%	10%	5%	15%	26%	41%	39
8. Sustainable, constant pace	8%	10%	8%	15%	31%	28%	39
9. Technical excellence/good design	0%	10%	8%	13%	41%	28%	39
10. Simplicity	5%	13%	5%	18%	33%	26%	39
11. Self-organising teams	3%	13%	8%	23%	26%	28%	39
12. Continuous improvement	0%	4%	7%	15%	41%	33%	27

Table 9: Frequency distribution of importance degree of agile principles

The mean score, mode and variance of importance degree per principle is also calculated to get overall and comparable insight. As Table 10 indicates, the mode for most principles are 6 which means most selected option was ‘very important’ by the respondents.

Agile principles	Mean	Mode	Variance
1. Customer satisfaction/outcome	5.18	6	1.05
2. Welcome change	4.77	6	1.71
3. Deliver frequently with shorter timescale	5.08	6	0.97
4. Business and developers' daily collaboration	4.56	6	1.83
5. Support and trust motivated individuals	4.87	6	1.80
6. Face-to-face communication	4.90	6	1.94
7. Working software as a measurement of progress	4.74	6	2.09
8. Sustainable, constant pace	4.36	5	2.55
9. Technical excellence/good design	4.69	5	1.59
10. Simplicity	4.38	5	2.30
11. Self-organising teams	4.41	6	2.09
12. Continuous improvement/retrospectives	4.93	5	1.15

Table 10: Mean score and mode for each agile principle

4.2. Mapping results and analysis

The mapping of agile principles and practices was sent to 25 agile experts and academicians from whom 6 of them have entirely completed the mapping. This response rate is sufficient to make decent conclusions on correlation between practices and principles. As this mapping is intended to be done by academic and field experts, there were no need of large number respondents; a few expert opinions were enough to make sound conclusions as a theoretical base. For the mapping itself, it was required to put a mark in the matrix of practice and principles where respondent considered that certain practice possibly supports/based on certain principle. The experts had differing views on mapping itself: Some saw the practices as means and principles as main goals of agile process; for them, practices are the tools/means to achieve, to realise the agile principles. Others think that if well-implemented, every practice should contribute to achieve all agile principles; they think that agile principles are implemented through the practices.

The analysis of this mapping is done as following: All mapping choices from 6 responses are brought together in one document in order to see the final result of matching. According to the numbers of vote, the percentage has been calculated as following: 1 vote – 17%, 2 votes – 33%, 3 votes – 50%, 4 votes – 67%, 5 votes-88%, 6 votes -100%. Accordingly, next categorisation is made for grouping the practices per principle:

- 17% – barely support
- 33-50% – moderate support
- 67-88% – strongly support
- 100% – exceptional support

For example, if one practice got 88% votes for certain principle, it means 5 experts considered that it matches that certain principle, and this practice strongly supports/based on given principle. Table 11 presents the mapping of principles and practices; the principles are given by numbers to save the complications and space. The results were calculated in percentages and colour-coded in an attempt to create easily readable results.

Practices/Principles mapping	1	2	3	4	5	6	7	8	9	10	11	12	Average rate per practice
Acceptance tests/functional test	67%	17%	33%	50%	0%	33%	33%	0%	17%	0%	0%	0%	21%
ATDD (acceptance test driven development)	67%	67%	33%	33%	0%	0%	17%	17%	33%	17%	17%	0%	25%
Automated build	67%	50%	67%	0%	0%	0%	67%	33%	33%	0%	0%	0%	26%

Backlogs (Product and Sprint)	0%	67%	17%	50%	33%	33%	0%	0%	0%	17%	17%	17%	21%
Backlog grooming	33%	83%	33%	33%	17%	33%	0%	17%	17%	0%	17%	17%	25%
Behavior-driven development (BDD)	33%	33%	0%	33%	17%	17%	17%	17%	17%	0%	0%	0%	15%
Burndown	50%	50%	33%	33%	17%	0%	67%	17%	17%	0%	17%	17%	27%
Collective code ownership	17%	0%	33%	0%	50%	33%	0%	0%	67%	0%	67%	0%	22%
Continuous integration	67%	33%	83%	33%	17%	0%	17%	67%	33%	0%	17%	0%	31%
CRC Cards	17%	17%	17%	0%	17%	67%	0%	0%	33%	0%	33%	17%	18%
Continuous deployment	67%	100%	83%	0%	17%	0%	33%	50%	50%	33%	0%	0%	36%
Daily meeting (daily scrum or stand up)	0%	50%	0%	50%	50%	67%	0%	17%	0%	0%	33%	33%	25%
Definition of done	83%	17%	50%	17%	0%	17%	67%	33%	33%	33%	33%	0%	32%
Definition of ready	50%	33%	83%	33%	0%	17%	17%	17%	33%	17%	17%	0%	26%
Estimation	0%	0%	17%	67%	33%	50%	0%	33%	0%	0%	17%	33%	21%
Exploratory testing	17%	50%	17%	33%	33%	17%	33%	17%	17%	17%	0%	0%	21%
Facilitation	17%	0%	17%	50%	50%	50%	0%	17%	17%	0%	33%	17%	22%
Frequent releases	67%	50%	50%	0%	33%	0%	33%	0%	17%	17%	0%	0%	22%
Given-when-then	0%	0%	0%	50%	17%	33%	33%	0%	0%	0%	0%	0%	11%
Heartbeat retrospective (sprint/iteration retrospective)	17%	0%	0%	0%	33%	50%	17%	0%	17%	0%	33%	50%	18%
Information radiators	17%	17% 17%	17%	17%	33%	33%	0%	0%	17%	0%	17%	0%	14%
Integration	0%	17%	0%	0%	33%	17%	33%	0%	33%	0%	0%	0%	11%
Invest acronym INVEST	0%	0%	17%	17%	0%	17%	0%	0%	33%	0%	0%	0%	7%
Iteration/sprint	50%	33%	50%	33%	17%	0%	33%	17%	0%	0%	0%	17%	21%

Iterative development	17%	50%	33%	33%	0%	17%	33%	17%	0%	17%	0%	0%	18%
Incremental development	0%	17%	67%	17%	17%	17%	50%	17%	0%	0%	0%	0%	17%
Kanban board	0%	33%	33%	17%	33%	33%	17%	17%	0%	17%	33%	0%	19%
Lead time/cycle time	50%	0%	67%	17%	0%	0%	17%	33%	0%	17%	17%	0%	18%
Milestone retrospective	17%	0%	0%	50%	0%	17%	0%	0%	0%	0%	0%	50%	11%
Mock objects	0%	0%	17%	0%	0%	0%	17%	0%	33%	0%	17%	0%	7%
Niko Niko calendar (mood board)	17%	0%	0%	0%	33%	33%	0%	17%	0%	0%	17%	17%	11%
Pair programming	0%	0%	17%	0%	17%	33%	0%	17%	33%	17%	17%	0%	13%
Personas	50%	0%	0%	33%	0%	0%	0%	0%	0%	0%	0%	0%	7%
Points (estimate in)	0%	33%	0%	0%	17%	33%	0%	17%	0%	17%	17%	0%	11%
Planning poker	0%	33%	0%	0%	0%	33%	0%	17%	0%	17%	17%	0%	10%
Project chartering	17%	17%	33%	33%	0%	33%	17%	0%	17%	0%	17%	0%	15%
Quick design session	17%	0%	0%	17%	17%	17%	17%	0%	33%	17%	33%	17%	15%
Refactoring	0%	0%	17%	0%	0%	0%	0%	17%	67%	33%	17%	0%	13%
Relative estimation	0%	0%	17%	17%	17%	33%	0%	0%	0%	0%	17%	0%	8%
Role-feature reason	0%	0%	0%	17%	17%	33%	0%	17%	0%	0%	0%	0%	7%
Rules of simplicity	0%	0%	0%	17%	0%	0%	0%	0%	17%	50%	33%	17%	11%
Scrum of scrums	17%	17%	17%	17%	0%	17%	0%	0%	0%	0%	17%	0%	9%
Simple design	17%	0%	0%	0%	0%	0%	17%	0%	0%	50%	17%	0%	8%
Sign up for tasks	17%	17%	0%	17%	50%	17%	0%	17%	0%	17%	33%	0%	15%
Story splitting	33%	17%	17%	17%	0%	0%	17%	17%	0%	33%	0%	0%	13%
Story mapping	17%	17%	0%	0%	0%	0%	17%	0%	0%	17%	0%	0%	6%
Sustainable pace	17%	17%	0%	17%	17%	0%	0%	33%	0%	0%	0%	0%	8%
Task board	17%	17%	33%	0%	17%	50%	17%	0%	0%	0%	17%	0%	14%

Test-driven development (TDD)	33%	0%	33%	0%	0%	0%	33%	0%	50%	0%	0%	0%	12%
Team (whole)	17%	0%	0%	17%	50%	17%	17%	17%	0%	0%	33%	0%	14%
Team room	33%	17%	0%	17%	17%	67%	0%	0%	0%	0%	33%	0%	15%
Three C's	17%	17%	0%	17%	0%	50%	0%	0%	0%	17%	17%	0%	11%
Three questions	0%	50%	0%	17%	17%	33%	0%	0%	0%	33%	17%	17%	15%
Time-box	17%	33%	17%	0%	17%	0%	17%	17%	0%	17%	0%	0%	11%
User stories	17%	33%	17%	17%	0%	0%	17%	17%	0%	17%	0%	0%	11%
Usability testing	33%	33%	33%	17%	0%	17%	0%	17%	17%	0%	0%	0%	14%
Ubiquitous language	17%	0%	17%	33%	0%	33%	0%	0%	33%	0%	17%	0%	13%
Unit testing	33%	0%	17%	17%	0%	0%	50%	0%	50%	0%	0%	0%	14%
Velocity	17%	17%	17%	17%	0%	0%	0%	17%	0%	0%	17%	17%	10%
Version control	17%	0%	33%	17%	0%	0%	17%	0%	50%	0%	17%	0	13%
Average rate per principle	23%	21%	22%	19%	15%	20%	15%	12%	16%	9%	14%	6%	

Table 11: Practice support for each agile principle

Table 11 indicates that some practices support certain principles very strongly, while some are barely based on any principle. There is no single practice that doesn't support any of 12 principles, indicating that all the listed practices are more or less based on at least one agile principle, which is itself a positive finding. On the other hand, the choices of experts strongly differ and therefore make it difficult to make final conclusions. However, some practices received more vote than the others. Initial impression is that first list of practices are more supportive than the practices placed in other sheets; one possible explanation could be that respondents were more enthusiast in the beginning and then got bored of this long list of practices and have done it more quickly at the end.

According to the table 11, Principle 1, which reads as "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software", is strongly supported by the following practices: acceptance tests, ATDD, automated build, continuous integration, continuous deployment, definition of done, and frequent releases. The practices that moderately support Principle 1 are backlog grooming, BDD, burndown chart, Definition of ready, Iteration/sprint, Lead time/cycle time, Personas, Story splitting, Test-driven development (TDD), Usability testing, and Unit testing.

Moreover, Principle 2, which reads as “Welcome changing requirements, even late in development; Agile processes harness change for the customer's competitive advantage”, is strongly supported by the practices of ATDD, Backlogs, Backlog grooming. Alternatively, moderate support is given by Automated build, BDD, Burndown Chart, Continuous integration, Daily meeting, Definition of ready, Exploratory testing, Frequent releases, Information radiators, Iteration/sprint, Iterative development, Kanban board, Points (estimate in), Planning poker, Three questions, Time-box, User stories, and Usability testing to Principle 2. The only practice/principle match in entire mapping that got exceptional support level is Continuous deployment / Principle 2. All experts agreed on the claim that practice of continuous deployment supports Principle 2.

On the other hand, Principle 3, which reads as “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale”, is strongly supported by Automated build, Continuous integration, Continuous deployment, Definition of ready, incremental development, and Lead time/cycle time. Meanwhile, moderate support level is achieved by Acceptance tests/functional test, ATDD, Backlog grooming, Burndown Chart, Collective code ownership, Definition of done, Frequent releases, Iteration/sprint, Iterative development, Kanban board, Project chartering, Task board, TDD, and Usability testing. All these practices enable to deliver working software frequently with shorter timescale.

According to the mapping, Principle 4, which reads as “Business people and developers work together daily throughout the project”, is achieved through the implementation of following practices: surprisingly, there is no practice that strongly supports this principle. Moderate support level for this principle is found in Acceptance tests/functional test, ATDD, Backlogs, Backlog grooming, BDD, Burndown Chart, Continuous integration, Daily meeting, Definition of ready, Estimation, Exploratory testing, Facilitation, Given-when-then, Iteration/sprint, Iterative development, Milestone retrospective, Personas, Project chartering, and Ubiquitous language.

Also for Principle 5, which reads as “Build projects around motivated individuals; give them the environment and support they need, and trust them to get the job done”, there are also no practices which support this principle strongly. Backlogs, Collective code ownership, Daily meeting, Estimation, Exploratory testing, Facilitation, Frequent releases, Heartbeat retrospective, Information radiators, Integration, Kanban board, Niko Niko calendar, Sign up for tasks, and Team (whole) support this principle at a moderate level. It is remarkable that most of these practices got 3 votes and are indeed very useful for building motivated teams, while supporting and trusting them.

Next, there are only two practices that strongly support Principle 6, which reads as “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation”, and they are Daily meeting and Team room. Otherwise, moderate support for face-to-face conversation principle comes from Acceptance tests, Backlogs, Backlog grooming, Collective code ownership, CRC Cards, Estimation, Facilitation, Given-when-then, Heartbeat retrospective, Information radiators, Kanban board, Niko Niko calendar, Pair programming, Points (estimate in), Planning poker, Project chartering, Relative estimation, Role-feature reason, Task board, Three C’s, Three questions, and Ubiquitous language. Furthermore, Principle 7, which reads as “Working software is the primary measure of progress”, is strongly supported only by practice of Definition of done. However, many practices support it at a moderate level, such as Acceptance tests, Automated build, Burndown Chart, Continuous deployment, Exploratory testing, Frequent releases, Given-when-then, Integration, Iteration/sprint, Iterative development, incremental development, TDD, and Unit testing.

For Principle 8, which reads as “Agile processes promote sustainable development; the sponsors, developers and users should be able to maintain a constant pace indefinitely”, no strongly supporting practices is found. Alternatively, moderate support is achieved by Automated build, Continuous integration, Continuous deployment, Definition of done, Estimation, Lead time/cycle time, Sustainable pace. On the other hand, Principle 9, which reads as “Continuous attention to technical excellence and good design enhances agility”, is strongly supported quite logically by Refactoring. Moderate support for this principle is given by ATDD, Automated build, Collective code ownership, Continuous integration, CRC Cards, Continuous deployment, Definition of done, Definition of ready, Integration, Invest acronym INVEST, Mock objects, Pair programming, Quick design session, TDD, Ubiquitous language, Unit testing, Version control.

According to the mapping, Principle 10, which reads as “Simplicity the art of maximizing the amount of work not done is essential”, has no strong support from any of the practices. however, it is moderately supported by Continuous deployment, Definition of done, Refactoring, Rules of simplicity, Simple design, Story splitting, Three questions. Principle 11, which reads as “The best architectures, requirements and designs emerge from self-organizing teams”, also doesn’t have any strongly supporting practices. Daily meeting, Definition of done, Facilitation, Heartbeat retrospective, Kanban board, Quick design session, Rules of simplicity, Sign up for tasks, Team (whole), Team room provide moderate support. Finally, Principle 12, which reads as “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly”, is moderately supported by Daily meeting, Estimation, Heartbeat retrospective, Milestone retrospective.

The further analysis is done to find out the average rate of each principle in order to determine how supportive they are in general and it is displayed also in Table 11. However, it should be kept in mind that this measurement can be misleading as sometimes one principle strongly supports one or two principle which is quite logical and normal, while the average rate includes other unsupported principles also into account. Therefore, such a high level of deviation doesn’t give accurate picture. Nevertheless, as Table 11 indicates, most supportive practices are ATTD, automated build, backlog grooming, burndown, collective code ownership, continuous integration, continuous deployment, daily meeting, definition of done, definition of ready, and facilitation. Moreover, according to average rate per principle, most supported principles are 1,2,3 and 6. And least supported are 8,10, and 12.

5. DISCUSSION

In this chapter, the results of survey and mapping will be further discussed, analysed and compared with the secondary data from other surveys to validate and back up own primary data. As the descriptive statistics has been used to analyse the survey and mapping data, the discussion of the results is also limited and cannot be generalised or can be named as confidently reliable due to the fact that no significance tests and inferential statistics analysis has been made. It should be noted that the mean scores, which is the main estimate used for central tendency analysis, is not tested for significance and therefore carries limited reliability. Therefore, the analysis results should be interpreted with caution. Moreover, the possible conclusions and answers gained from the analysis are extremely tentative given the limited analysis depth.

Next, the classification of practices according to AgileAlliance mapping (AgileAlliance) is used to discuss the practice implementation results. In addition, principle classification made by the author is employed to reveal which group of principles are most supported by the practices. Furthermore, the attempt is made to find out the actual presence degree of agile principles in agile software development process by analysing both survey and mapping results together.

5.1. Discussion of practice implementation degree

The present study examined the implementation degree of agile practices. The bar chart of mean scores of implementation degree per agile practices presented in Figure 11 makes it more visible to observe the differences and compare them. It would seem that substantial differences exist among practices on implementation degree.

Before examining the mean score comparison, it is interesting to know which practices received the most “not applicable” votes. According to Table 8, possibly BDD, CRC Cards, Given-when-then, Invest, Kanban board, Milestone retrospective, Niko Niko calendar, Lead/cycle time, Project chartering, Rules of simplicity, Scrum of scrums, Three C’s, Ubiquitous language are the practices that got more than 15% vote (the most being 34% for Scrum of Scrums) for ‘not applicable’ option meaning that they are just not applicable to respondents agile processes.

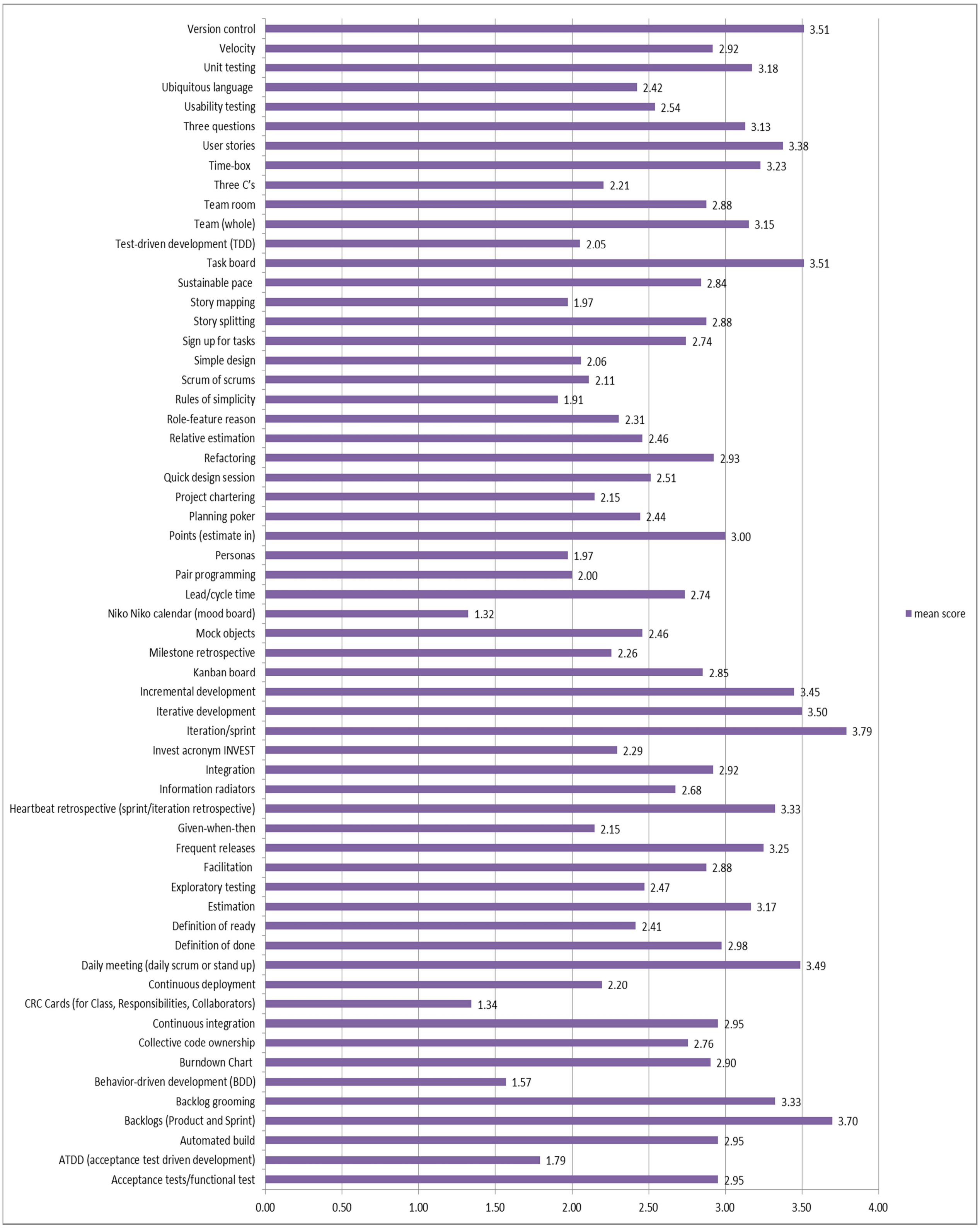


Figure 11: Mean scores of implementation for each agile practice

Based on Figure 11, the possible list of top 10 agile practices with the highest mean score (mostly <=3.5) could be suggested displayed in Table 12.

Practice	Implementation mean score
1. Iteration/sprint	3.79
2. Backlogs	3.70
3. Task board	3.51
4. Version control	3.51
5. Iterative development	3.50
6. Daily meeting	3.49
7.Incremental development	3.45
8. User stories	3.38
9.Heartbeat retrospectives	3.33
10. Backlog grooming	3.23

Table 12: Top 10 agile practices with the highest mean scores

These data seem to show that the short iterations, backlogs, task boards, version control, iterative development and daily meetings are probably the most widely practiced and almost fully implemented by most respondents. It is worth noting that, the presence of version control in this list is somewhat surprising as it is not an old classic of agile practices, yet is has gained much popularity in recent times. Otherwise, the results are likely to be in line with popular agile practices suggested in literature review. Alternatively, the list of the least implemented practices could also be articulated based on the Figure 11. Accordingly, the following practices received the lowest mean score of less than or about 2, which suggest that they are likely partly implemented or almost not implemented.

Practice	Implementation mean score
1. Niko Niko calendar	1.32
2. CRC cards	1.34
3. BDD	1.57
4. ATTD	1.79
5. Rules of simplicity	1.91
6. Personas	1.97
7. Story mapping	1.97
8. Pair-programming	2.0
9. TTD	2.05
10. Simple design	2.06

Table 13: List of practices with lowest mean scores

Table 13 indicates that Niko Niko calendar, CRC cards, BDD, ATTD, pair programming, TTD, and Simplicity are probably among the least implemented practices by the respondents. Some of them such as Niko Niko calendar, ATTD, CRC cards and Personas are less known agile practices and therefore also not widely used. On the other hand, practices such as pair programming, TTD, simple design which are the core XP practices also appears among the list of probably least implemented ones. As known from literature review, one of the most studied agile practices is pair programming and it may be speculated from the survey results that it is not much popular in the industry at all.

Equally interesting is to analyse variances in practice implementation degrees to measure how far implementation degree choices are spread out in order to detect the 'controversial' practices. Figure 14 illustrates variance measurement for each practice. It could be interpreted that, possibly the most consistent practices are Iteration/Sprint, Backlog, Iterative development, Task Board, Incremental development, Heartbeat Retrospectives, Backlog Grooming, Daily Meeting, frequent releases, User stories and Unit testing with variance score lower than 1 or equal to 1, which indicates that respondent choices for these practices tend to be close to each other and also to mean. Alternatively, the list of the most 'controversial' practices might include Rules of Simplicity, Scrum of scrums, CRC cards, three C's, Niko Niko calendar, BDD, Role-feature-reasons, Project chartering, ATTD, and Given-when-then with variance score of more than 2.20, which indicates that the choices were very spread out from each other and from the mean.

Interestingly, if the mean score and variance score figures are compared for each practice, possible correlation could be found between them. Consequently, the practices with highest mean score also turn out to be the most consistent ones. On the contrary, the practices which have lowest mean score have also highest variance degree which makes them more 'controversial' practices.

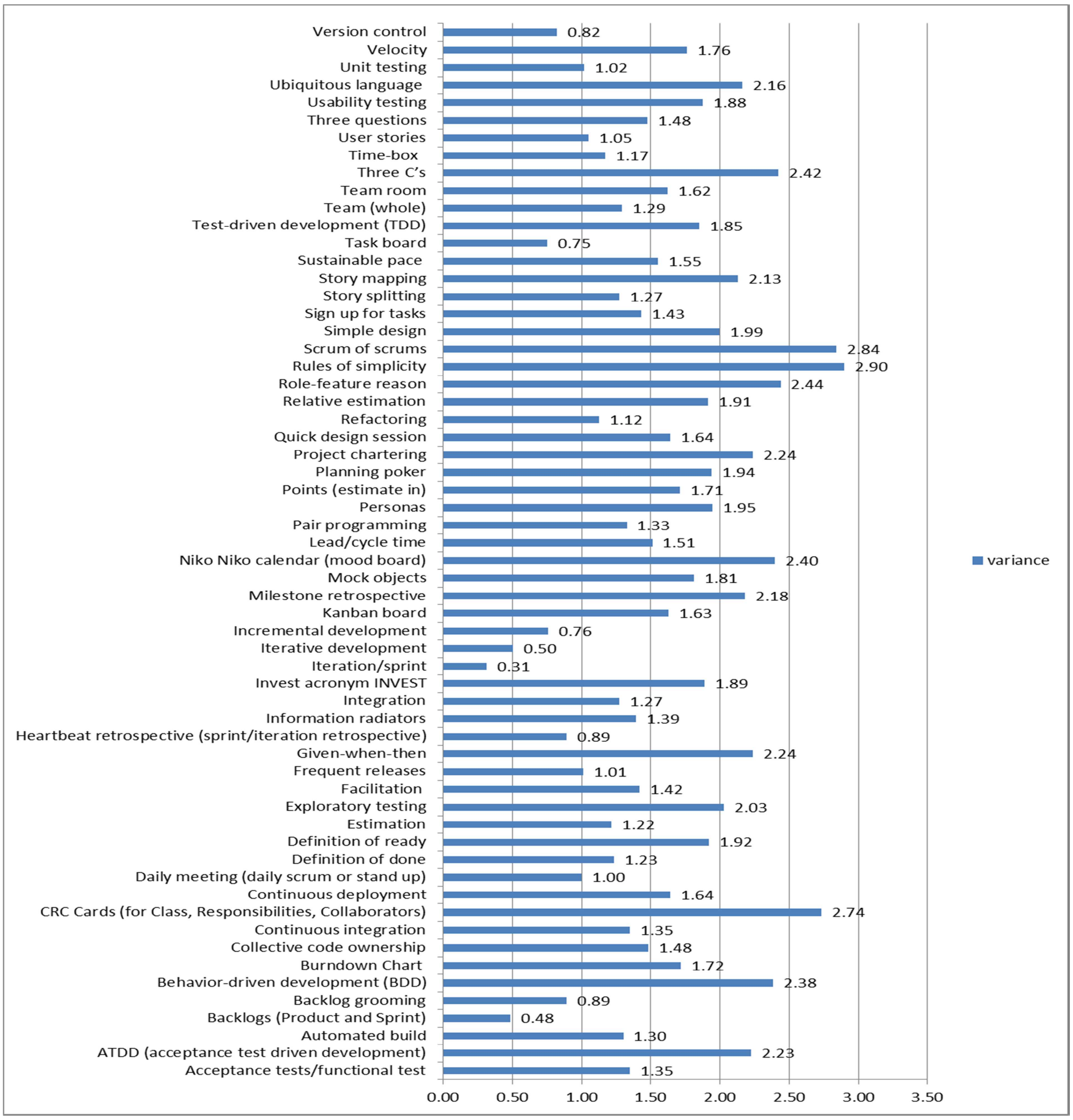


Figure 12: Variance of practice implementation degree

Subsequently, next step is to compare the thesis survey data with secondary data mentioned in chapter 3 in order to back it up and possibly validate. As stated before, VersionOne's 'State of Agile' survey has its own list of most practiced agile techniques, and they are: Daily stand-up, Short iterations, Prioritized backlogs, Iteration planning, Retrospectives, Release planning, Unit testing (VersionOne, 2015). Although this list presents some variations, they are comparable with the present survey results, as most of them are implemented by respondents with high mean scores. Alternatively, the least practiced techniques according to VersionOne's survey are Behaviour-

Driven Development, Agile games, Pair programming, Continuous deployment, Automated acceptance testing, Collective code ownership, Story mapping, Kanban, and Test-Driven Development (TDD) (VersionOne, 2015). Because less number of agile practices (25) is used for the survey of VersionOne, it doesn't cover all agile practices. However, some least popular agile practices of VersionOne are indeed have lower mean score in thesis survey results, such as Planning pocker, Continuous deployment, and some of them are even present in top 10 lowest implemented practice list, such as BDD, TDD, Story-mapping and Pair programming, which makes these two surveys quite comparable. Furthermore, it might be the case that VersionOne survey could validate the results of thesis survey to a certain extent by reinforcing its results once more. Interestingly, one practice that requires special attention is Kanban; while in VersionOne survey it has been ranked as one of the least practiced ones, in thesis survey it is among relatively high implemented practices with mean score of 2.85 meaning that it is mostly implemented by the respondents. It may be speculated that a possible explanation for this is the rising popularity of Kanban and Lean methods in recent times.

Another survey, which is conducted by Williams, defines the top 5 most essential agile practices as following: Short iterations, Continuous integration, "Done" criteria, Automated tests are run with each build, Automated unit testing (Cohn, 2010). In her first survey, Williams didn't ask for implementation degree, but only how essential is the practice for agility. That is why it is not much comparable with the results of present survey, although it gives another overview of practices. For example, it might be articulated that above listed practices are indeed implemented with high mean scores according to the figure 11. Additionally, in her CA assessment survey she asked the degree of adoption of different agile practices although with different naming and definitions than in the thesis survey. As mentioned before, all agile statements/practices were grouped under 7 dimensions such as requirements, technical practices, quality, planning and so on (Williams L. R., 2010). If we compare the mean scores of each group of practices to the thesis survey results, they could presumably be called compatible. For example, in William's survey most highly adopted practice groups were requirements and knowledge creating (Williams L. R., 2010); in thesis survey most implemented practices with highest mean scores also possible could belong to these categories such as short iterations, backlogs, task boards, daily meetings, iterative/incremental development, version control and so on. On the other hand, lowest adopted practice groups were technical and quality practices according to William's survey (Williams L. R., 2010); and thesis survey suggests also similar results, for example BDD, CRC cards, ATDD, TTD, pair programming, simplicity, etc. Hence, it might be the case that the results of two surveys are considerably compatible with each other if it is compared at high level.

The last survey that could be used to back up the survey results on implementation degree of practices is AmbySoft survey. When the results of Ambysoft survey is reflected to RQ1, it first should be noted that as average rating calculation and practice naming is slightly different in two surveys, the results are also less comparable. Nevertheless, the highest rated practices in Ambysoft survey are: Iteration planning (3.54), Daily Scrum Meeting (3.29), Prioritized worklist (3.08), High-level release planning (2.19), Coding Standards (2.30), Collective Code Ownership (1.97), Active Stakeholder Participation (1.95), Continuous integration (1.94), Retrospectives (1.84) and Code Refactoring (1.79) (Ambler S. V., 2008). Several practices could be found similar to the thesis survey's highly implemented practices, such as backlogs-prioritised worklist and daily meetings. If looked for the least commonly used practices in Ambysoft survey: Pair programming (-1.34), Automated Acceptance Testing (-0.87), Database regression testing (-1.03), Executable Specs (-1.43), TDD (-0.08) (Ambler S. V., 2008). It is worth noting that only TTD and pair programming appears in thesis survey results of least implemented practices. The summary of comparison of 4 surveys can be brought together in Table 14. It is worth noting that the calculation of each rate and percentage is different in every survey and therefore only the sequence and the practice names can be compared, neither the scores nor rates:

Thesis survey (2015)	**VersionOne survey (2014)**	**Ambysoft Survey (2008)**	**Survey Williams(2010)**
Most practiced/implemented practices			
1. Iteration/sprint-3.79 2. Backlogs -3.70 3. Task board - 3.51 4. Version control- 3.51 5. Iterative development - 3.50 6. Daily meeting - 3.49 7. Incremental development-3.45 8. User stories- 3.38 9. Heartbeat retrospectives-3.33 10. Backlog grooming - 3.23	80% Daily stand-up 79% Short iterations 79% Prioritized backlogs 71% Iteration planning 69% Retrospectives 65% Release planning 65% Unit testing 56% Team-based estimation 53% Iteration reviews 53% Task board	Iteration planning (3.54) Daily Scrum Meeting (3.29) Prioritized worklist (3.08) High-level release planning (2.19) Coding Standards (2.30) Collective Code Ownership (1.97) Active Stakeholder Participation (1.95) Continuous integration (1.94) Retrospectives (1.84) Code Refactoring (1.79)	1. Short iterations (30 days or less) 2. Continuous integration 3. “Done” criteria 4. Automated tests are run with each build 5. Automated unit testing
Least practiced/implemented practices			
1.Niko Niko calendar-1.32 2. CRC cards-1.34 3. BDD-1.57 4. ATTD - 1.79 5. Rules of simplicity-1.91 6. Personas-1.97	9% Behaviour-Driven Development (BDD) 13% Agile games 21% Pair programming 24% Continuous	Pair programming (-1.34), Automated Acceptance Testing (-0.87), Database regression testing (-1.03), Executable Specs (-1.43),	

7. Story mapping- 1.97 8. Pair-programming-2.0 9. TTD-2.05 10. Simple design-2.06	deployment 24% Automated acceptance testing 27% Collective code ownership 29% Story mapping 31% Kanban 34% Test-Driven Development (TDD) 36% Refactoring	TDD (-0.08).	

Table 14: Comparison of the results of all surveys on agile practices

The more high level analysis of practices by grouping them into different clusters delivers another set of valuable results. As it is mentioned in previous chapters, the AgileAlliance has mapped all 60 practices in 6 different categories. Some of them are agile methods, while others are development phases: Extreme Programming, Scrum, Lean, Teams, Product management, Devops, Design, Testing, and Fundamentals (AgileAlliance). The full mapping is shown in literature review chapter. Accordingly, the mean score for each category could be calculated to see the possible comparative implementation degree of each separate category. This comparison is presented in Figure 13.

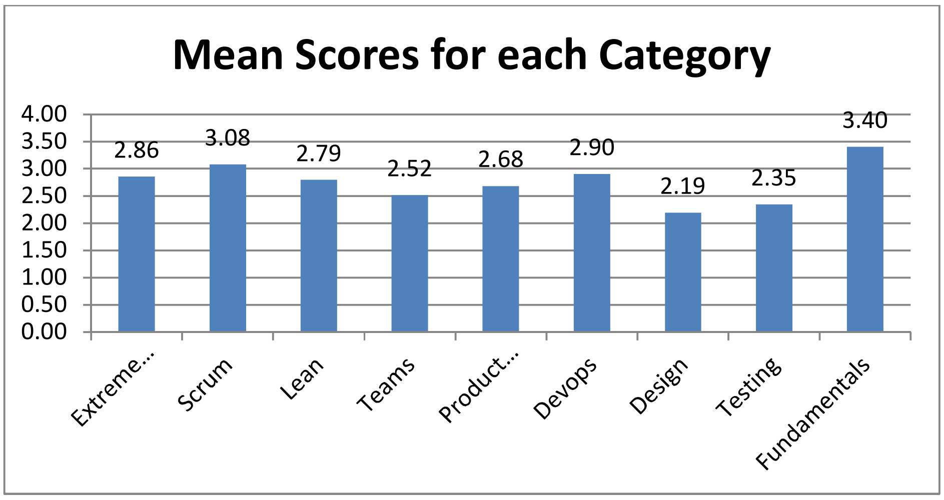


Figure 13: Implementation mean scores for each practice category

The data seem to show that the highest implementation degree might be observed in practice group of Fundamentals which includes 3 core agile practices: incremental development, iterative development and version control. The next highest score belongs to Scrum cluster. This finding reinforces the view that Scrum is the most used agile method (VersionOne, 2015) and therefore its practices are also widely implemented in the industry. Third most implemented group of practices seems to be Devops, which is very new but fast growing agile method. Interestingly, it is even not mentioned in any academic literature and obviously under researched. However, being possibly the third highest implemented group of practices and presumably scoring even slightly higher than the old classic agile method XP suggest that Devops are getting popular in the industry. The lowest implementation score seems to belong to the Design and Testing cluster. Those technical practices are probably the least implemented ones among the respondents; likewise this fact is also supported by the survey results of Williams.

Equally interesting is the analysis of implementation mean score data with regard to Meyer's critical classification of practices which has been elaborated in the chapter 2. Table 15 suggests that the implementation degree of each category correlates with its usefulness criteria of Meyer. Accordingly, the least implemented practice group is 'bad' practices, while the most implemented one is 'brilliant' practices which Meyer considers the most useful and novel contribution of ASD to the software development world. Meyer might be content to hear these results as it suggests that the industry indeed has chosen and implement the most useful practices.

Meyer's practice classification	**Mean score**
'bad' practices	2.47
'hype' practices	2.59
'good' practices	3.21
'brilliant' practices	3.29

Table 15: Implementation degree of practice classification of Meyer

5.2. Discussion of principle importance degree

Generally, the survey data seem to suggest that all principles are considered important by the respondents, as the most of the mode for the principles are either 5 or 6 which means important or very important. Possibly 8 of 12 principles seem to be considered important/very important by only more than 60% of respondents which itself is not so high percentage compared to William's survey where she claimed that 11 of 12 principles had at least 80% of the respondents giving the principle a rating of 4 or 5 (out of 1-5 scale) (Hastie, 2010). According to the survey results, the following mean scores of importance per principle is calculated. It is worth reminding that the mean score is not tested for significance and therefore carries limited reliability.

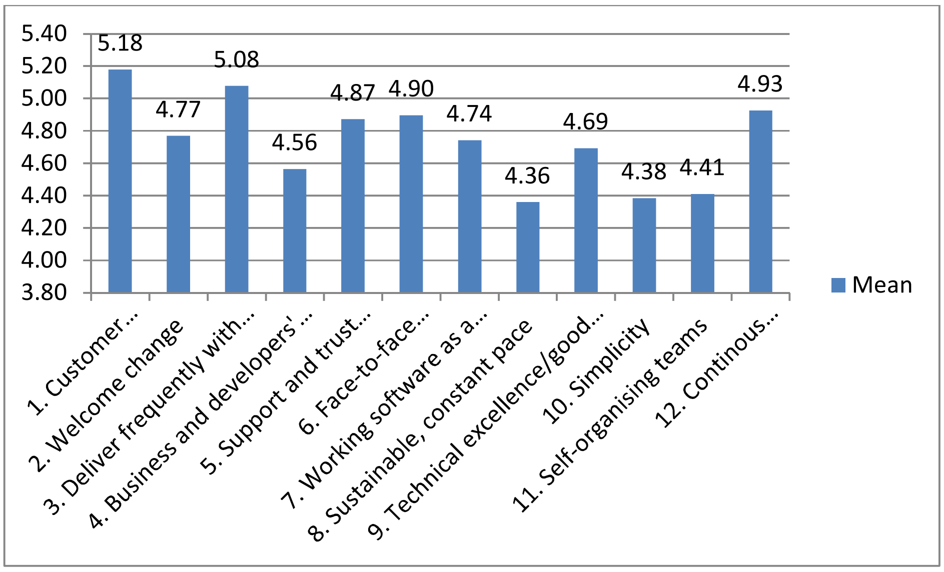


Figure 14: Mean scores of importance degree for each agile principle

As figure 14 suggests, mean scores fluctuate between 5.18 and 4.36 which means between 'partly important' and 'important'. It might be the case that the respondents consider principle 1 and 3 possibly as the most important agile principles. What this seems to indicate is that respondents value customer satisfaction and frequent delivery the most in what we call agile. The third most important principle might be 12 on continuous improvement/retrospectives, and the next one could be principle 6 on face-to-face communication. It should be noted that the last principle is evaluated by fewer respondents than the others and therefore the higher mean score here can be misleading. Therefore, it may be speculated that the principle 6 might be actually the third most important one, and principle 5 on motivated individuals could possibly be the fourth. Alternatively, data seems to show that principles on sustainable pace (8), simplicity (10), and self-organizing teams/emergence (11) are considered least important ones by the respondents. Interestingly, principle 2 on welcoming change seems to have middle importance for respondents since it hold 6th place on importance ranking. This is somewhat surprising because from the literature review it has been observed that the one of the most essential feature of agility was its ability to react change quickly, while it might be the case that practitioners don't value this feature as it should be. Instead, satisfying customer, delivering frequently and continuous improvement and/or face-to-face communication are possibly the most valued agile features and principles.

If the thesis survey results are compared with the William's first survey, it is observed that her list of top five important principles corresponds to the results of present survey. According to her survey, the most important principles were on customer satisfaction (principle 1), working software (7), deliver frequently (3), motivated individuals (5), and continuous improvement/retrospectives (12) (Cohn, 2010). The only principle that deviates from the thesis survey results is principle 7 on working software. Overall, the thesis survey results on principle importance reflect William's survey. Additionally, it is interesting to analyse the principle importance data according the classification of principles mentioned in the literature review chapter. If the mean scores per category are calculated, it would seem that the highest importance rate presumably belongs to working software and customer satisfaction related principle groups. The lowest importance could be given to quality/technical related principles. Table 16 presents the whole list of categories and corresponding mean scores.

Principle groups	Mean scores
Team/organizational	4.69
Working software	5.00
Collaboration	4.79
Quality/technical	4.60
Customer satisfaction	4.97

Table 16: Mean scores of importance per principle category

5.3. Discussion of the presence of principles in agile development

Considering that the practice implementation degree, principle importance degree and principle/practice mapping has been analysed, the next step is to find out the presence of agile principles in ASD process. It could be done through: first, by determining the list of practices that most supports each agile principle from the mapping; and then, by calculating the mean score for each principle to define how principles are practiced. Finally, the principle importance degree mean scores will be compared the principle practicing mean scores to determine if they correspond to each other. One should keep in mind that, all those calculations are done without significance testing and are therefore highly tentative and with limited interpretation possibilities.

In this regard, the list of most supportive practices per principle has been determined based on mapping table in analysis chapter and the same list is used to calculate the principle practicing mean score per principle. It is vitally important to note that when there is strongly supporting practices, only they were taken into account to calculate the mean score, otherwise only moderate supporting practices per principle have been collected for mean score calculation. Therefore, sometimes if there is only a few strongly supporting practices, the mean score can be high and it is also misleading. Accordingly, the mean scores are calculated and presented in Figure 14.

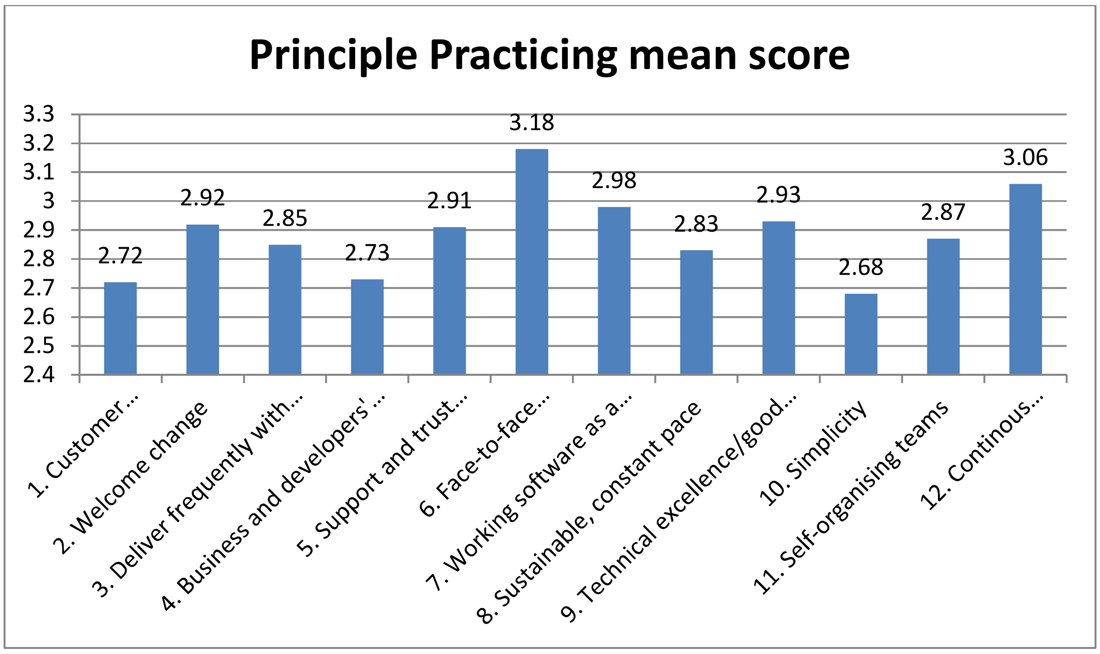


Figure 15: Average mean scores of agile practices supporting each agile principle – principle presence

These principle presence mean scores might be the actual proof of how far the principles are practiced through agile practices. The data displayed in Figure 15 seem to suggest that, possibly the most present principles in ASD process are the ones on face-to-face communication (6), on continuous improvement (12) and on working software (7) according to the calculated mean scores. On the other hand, it is likely that the respondents adhere to the principles of 1, 4, and 10 the least of all. These results could be analysed per category groups of principles mentioned above to gain more high level insight. As Table 17 displays, although mean scores are very close to each other, the analysis seems to show that the most practiced group of principles might be collaboration/communication and team/organizational related ones. The least adhered principle cluster is likely to be customer satisfaction related principles which suggests that possibly these principles are used less than others.

Principle groups	**Mean scores**
Team/organizational	2.970
Working software	2.85
Collaboration/communication	2.973
Quality/technical	2.863
Customer satisfaction	2.82

Table 17: Mean scores of principle presence per category

5.4. Discussion of perceptions vs. realities on agile principles' importance and presence

Afterwards, the principle importance mean scores are included into the analysis in order to make an attempt to compare the perceptions and the reality. However, it is worth noting that the mean score for principle importance is calculated based on the 6-point Likert scale, while implementations mean score is based on 4-point Likert scale. This makes it difficult to do accurate comparisons based on numbers. Therefore, only the ranking order of principles in both of the mean scale charts might be compared and analysed. Table 18 illustrates this comparison.

Rank	Principle presence mean score ranking/sequence	Mean score	Principle importance mean score ranking	Mean score
1.	6. Face-to-face communication	3.18	1.Customer satisfaction/frequent delivery	5.18
2.	12. Continuous improvement/retrospective	3.06	3. Deliver frequently with shorter timescale	5.08
3.	7. Working software as a measurement of progress	2.98	12. Continuous improvement/retrospective	4.93
4.	2. Welcome change	2.92	6. Face-to-face communication	4.90
5.	5. Support and trust motivated individuals	2.91	5. Support and trust motivated individuals	4.87
6.	11. Self-organising teams	2.87	2. Welcome change	4.77
7.	3. Deliver frequently with shorter timescale	2.85	7. Working software as a measurement of progress	4.74
8.	9. Technical excellence/good design	2.84	9. Technical excellence/good design	4.69
9.	8. Sustainable, constant pace	2.83	4. Business and developers' daily collaboration	4.56
10	4. Business and developers' daily collaboration	2.73	11. Self-organising teams	4.41
11.	1. Customer satisfaction/frequent delivery	2.72	10. Simplicity	4.38
12.	10. Simplicity	2.68	8. Sustainable, constant pace	4.36

Table 18: Comparison of agile principles' importance degree with its presence degree

Table 18 might be helpful to grasp the differences in perceptions and reality on agile principles on a very high level and with tentative reservations. What the ranking comparison seems to indicate is that respondents' adherence to what they consider important might not be consistent at all. Analysis reveals that while they possibly give the most importance to the principle 1, the implementation of the same principle suggests the opposite results: the same respondents are likely to implement it in one of the lowest degree - it appears as 11th among 12 principles. The data indicates that the most widely practiced agile principle might be the one which is about face-to-face communication (6). The importance and presence of principle 12 could be comparable which suggests that respondents presumably adhere to this agile principle at the same degree as they claim it to be important. Another notable result belongs to principle 7. While it is considered not so highly important principle, it might be actually implemented with the third highest mean score of practices; that indicates higher implementation but middle importance. The data seem to show that the principle 2 on welcoming change is also more present in agile development process than it is considered to be important. On the other hand, Principles 5 and 9 hold the same place in both of the rankings which suggests that respondents adhere to these principles possibly at the same degree as they claim them to be important; in other words, their perceptions corresponds to their actual work process. Similar results might be stated for principles 4 and 10; respondents adhere to them more or less as strongly as they claim them to be important.

It is worth noting that, while principle 3 on frequent delivery is claimed to be the second most important principle, it is present in agile development process in less degree; that indicates lower presence but higher importance. Moreover, this result seems to be also inconsistent with the demographics results of survey where majority of respondents (about 90%) stated that they use short iteration/sprint (2-4 weeks) in their ASD, which indicates that they indeed follow the practices regarding principle 3. Therefore, it is somewhat surprising to conclude such low rate results. On the other hand, one possible explanation to this result can be the rising popularity of Lean/Kanban method whereby no iteration or sprint is used; instead continuous improvement is the central concept. The reactions of some respondents support this idea: they claimed that, in fact, there is no iteration in their process; they work with continuous-pull process where stories can take from couple of hours to weeks to complete.[15]

On the other hand, the practices that support principle 8 on sustainable pace is presumably implemented more frequently than they are considered to be important, the result is lower importance but higher implementation. Generally, this unique table of both implementation and importance degree of principles seems to provide quite remarkable insights on the issue of perceptions vs. reality. It reveals that, possibly only 5 principles have the same or similar

[15] Personal e-mail communication with respondents

corresponding implementation and importance mean scores. At least 3 principles (1, 3, and 7) are likely to have totally opposite places in presence and importance ranking which provide support for the view that respondents do not follow the practices that support the principles which they consider most important. The rest of the principles also seem to show variances but with less difference degree.

The more high level analysis could be discussed by comparing the presence/importance mean scores for principle categories specified before. As comparison of Tables 14 and 15 indicates, it might be the case that, while the most important principle groups were presumably working software and customer satisfaction related, the most practiced/present principle groups are appears to be totally different ones: collaboration/communication and team/organizational related principles. That reveals a refreshing and concerning view of reality, since it suggests that there is likely to be some missing link between perceptions and realities. It could be interpreted in different ways: First, as one respondent stated in personal communication, agile practitioners are not aware of agile principles; they apply just practices mechanically and don't know the behind values and principles, but consider themselves as agile teams. In fact, this argument is a concern of several respondents who argued that most agile teams have no agile mindset and go for only mechanical practices which they find interesting. They further claim that agile development should be first and foremost about the mindset and values of agile rather than a bunch of practices. [16] In fact, this view is also supported by Medinilla (2012) from literature review. Second, alternatively, it could be something experimental that deserves special attention for further research.

[16] Ibid.

6. CONCLUSION

In this chapter all works performed in this thesis is summarized by revisiting each of research question. It presents the observation drawn from the results by providing answers to the research questions. Hence, the analysis parts derived from the results are mainly used to define answers for the research objectives. Finally, the implications and limitations of this research are presented with focus on industrial practice and research, which includes future work in this area.

6.1. Research questions revisited

After analysing and discussing the primary and secondary data, the research questions could be now answered with better insights. The brief discussion of each research question follows as:

RQ1. What is the implementation degree of agile practices among practitioners?

The study indicates that implementation degree of agile practices vary with each practice, with the most widely implemented practices probably being Iterations, backlogs, iterative/incremental development, task boards, and version control. Possibly, the least implemented practices belong to technical and design practices and include Niko Niko calendar, CRC cards, Simplicity, BDD, TTD, Pair programming and others. Additionally, if it is analyzed from the AgileAlliance categories' point of view, the most implemented practice group is likely to be Fundamentals and Scrum, while the least implemented ones might be Design and Testing. Furthermore, Meyer's 'bad' practices scored lowest degree of implementation, while 'brilliant' ones seem to be the most implemented practice group among all 4 usefulness categories of Meyer. Finally, the results of thesis survey were compared with and backed up by the secondary data from other surveys and it has been observed that, overall, thesis study seems to reflect the previous findings with slight variances.

RQ2. How important are agile principles for practitioners?

The thesis survey results seem to show that merely 8 of 12 agile principles are considered to be important/very important by only slightly more than half of the respondents which might indicate the not so high level of support for agile principles among practitioners. Another possible explanation for the lower importance of principles may be that practitioners are not aware of them at all. This could reinforce the view that agile mindset turns out to be not the most important thing in the current industry practice. Moreover, the degree of importance differs per principle. Most important principles for practitioners are likely to be the principles on customer satisfaction, frequent delivery, face-to-face communication and continuous improvement (1, 3, 6, 12), whereas possibly the least important ones are principles 8, 10, 11. If analyzed through the lenses of categories, respondents seem to value the principles related to working software and customer satisfaction the most, while they presumably give less importance to principles related to quality/technical.

RQ3. What kind of correlation exists between agile practices and principles? In which degree do agile practices based on /support agile principles?

The correlational data would suggest that all agile practices listed in this thesis may support/be based on at least one agile principle. However, the degree of support is likely to vary greatly depending on the principle and practice. The table x presents it more visible with color-coded percentage of support. It is worth noting that possibly continuous deployment exceptionally supports the principle 2 on welcoming the change, which is the only principle/practice match that had 100% support vote. However, the implementation degree of this particular practice seems to be not so high (2.20 mean score) being near to 'partly implemented' range, while one would expect it to be one of the most implemented practices since it presumably supports so strongly one of the important principles. The list of practices presumably matching each principle is also discussed in detail in previous sections.

RQ4. Are agile principles, which are perceived to be important, also with the same degree actually present in agile software development process of practitioners through supporting practices?

This research question concludes all other research questions by using their results as an input for analysis. Hitherto the degree of implementation of agile practices is likely to be known, the perceived importance degree of agile principles is examined and the support degree of practices for each principle is discussed. What remains is to analyze them together to examine if the perceived importance level of principles is the same as the presence level of principles. Accordingly, the answer to this research question seems to be complicated. If it is evaluated per principle, answers 'yes' could be found for some principles which might suggest that for certain agile principles there is obviously overlap between the degree of principle importance and degree of principle practice; in other words, adherence to principles correspond to their importance level. Alternatively, if analyzed per category, unfortunately, more contradictory view seems to be found, hence predominately answers 'no' occur. That indicates possible inconsistencies, since the adherence to principles which is perceived to be important seems to be not in equal level: practitioners seem to follow in less degree what they find important. It is somewhat surprising that they possibly give importance to one group of principles, while the implementation degree of practices seems to indicate that totally another group of principles are more present in their ASD process. This potential mismatch is particularly important to reveal the possible dynamics in agile development process in terms of agile mindset/values vs. agile practices debate. As mentioned earlier in discussion part, it may be speculated that this is probably so because practitioners are likely to be less aware of principles and values of agile development and hence more busy with only applying the practices mechanically.

6.2. Limitations of this study

There were three main limitations in this study. First, the scope of the research is limited only to presence and importance of agile principles and therefore do not include other agile topics such as success factors, or the scaling of agile development. Furthermore, selected research methods come with their own limitations and concerns which have been discussed in chapter 3. The second limitation is that while quantitative research strategy makes it more possible to generalize the results to larger agile practitioners' groups, the low response rate of survey limits this possibility and becomes a threat to the study's validity. On the other hand, the survey data is supported by relevant secondary data from other related surveys and therefore acquires more validity in this sense. Finally, third limitation concerns the analysis of the collected data. As mentioned before, only descriptive statistics has been used to analyse the data, thereby without conducting any statistical significance tests and inferential analysis which on its turn may have influenced the broader and deeper analysis of the data. However, as it is discussed shortly, it opens up new opportunities for further research.

6.3. Implications for the industry and for further research

Two possible practical implications for the industry may emerge from this study. Most of the time it seems that practitioners don't implement the same principles that they find important and it creates concerns on the priorities and the missing link between the principles and practices. Clearly, steps should be taken to ensure that they consider better while choosing the practices, because sometimes what they choose doesn't support what they find important and they should better find out the exact practices they need as they consider important and then apply them in order to become agile. Additionally, the agile principle/value awareness should be more endorsed, as study results on inconsistencies between agile principle importance degree and presence degree may provide further evidence of the fact that value/mindset awareness might be the missing link in agile development process.

Alternatively, the study creates many thought-provoking opportunities for future academic research. There is few research conducted on ASD, and hardly any study on presence and importance of agile principles, therefore the further investigation on the topic could reveal various undiscovered issues. The thesis has paved the way for correlating agile principles and practices and for determining the perceived importance and the real presence of agile principles in agile software development process. Hence, this could be starting a point to further examining the subject. Indeed, a promising line of study would be to investigate the reasons behind such results gained in the thesis. Furthermore, the agile methods dimension of the issue can widen up the study and give more interesting insights with particular emphasis on method mixing, dynamics behind such mixture choices and so on. Since this is a first explorative study on this specific issue, it could be deepened and broadened further with more large pool of respondents and with more complicated analysis tools. Henceforth, the research could form the basis for subsequent longitudinal studies. Last but not least, the debate of principles vs. practices, agile mindset vs. agile practices is also a very intriguing issue to further research on. What are the implications: better first learn principles and then apply practices that truly support the core value of agile or just implementing some principles makes it enough to be agile? Further research could be conducted on this essential question.

7. Bibliography

Abrahamsson, P. C. (2009). Lots done, more to do': the current state of agile systems development research. *European Journal of Information Systems, 18*, 281-284.

Abrahamsson, P. S. (2002). *Agile software Development: Review and Analysis.* VTT.

Abrahamsson, P. W. (2003). New directions on agile methods: a comparative analysis. *Software Engineering*, 244 - 254.

Ågerfalk, P. a. (2006). Flexible and Distributed Software Processes: Old Petunias in New Bowls? *Communications of the ACM*, 27-34.

AgileAlliance. (n.d.). *Guide to Agile Practices*. Retrieved July 15, 2015, from AgileAlliance: http://guide.agilealliance.org/

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

Ambler, S. V. (2008). Agile Practices and Principles Survey. Retrieved June 16, 2015, from www.agilemodeling.com/surveys/

Barlow, J. e. (2011). Overview and Guidance on Agile Development in Large Organizations. *Communications of the Association for Information Systems, 29*.

Beck, K. ,. (2005). *Extreme Programming Explained: Embrace Change.* Boston: Addison-Wesley.

Bermejo, P. Z. (2014). Agile principles and achievement of success in software development: A quantitative study in Brazilian organizations. *Procedia Technology*, 718 – 727.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35(1)*, 64–69.

Boehm, B. T. (2003). *Balancing Agility and Discipline: A Guide for the Perplexed.* Boston: Addison-Wesley Longman Publishing Co.

Boehm, B. T. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, 30-39.

Bryman, A. (2004). *Social Research Methods.* New York: Oxford University Press.

Chow, T. C. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software, 81*, 961–971.

Coad, P. D. (1999). *Java Modeling in Color.* Englewood Cliffs, NJ: Prentice.

Cockburn, A. (2001). *Agile software development.* Boston: Addison-Wesley.

Cockburn, A. (2001). *Crystal Clear: A Human-Powered Software Development Methodology for Small Teams.* Reading, MA: Addison-Wesley.

Cockburn, A. (2007). *Agile Software Development: The Cooperative Game.* Addison-Wesley.

Cockburn, A. H. (2001). Agile software development: The business of innovation. *IEEE Computer*, 120-122.

Cohen, D. L. (2004). An Introduction to Agile Methods. *ADVANCES IN COMPUTERS*.

Cohen, D. L. (2004). An Introduction to Agile Methods. *Advances in Computers, 62*.

Cohn, M. (2010, June 17). What Does It Mean to Be Agile? Retrieved June 17, 2015, from https://www.mountaingoatsoftware.com/blog/what-does-it-mean-to-be-agile

Conboy, K. (2009). Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research*, 329 - 354.

Conboy, K. F. (2007). The Views of Experts on the Current State of Agile Method Tailoring. In *Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda* (pp. pp 217-234). US: Springer.

Couper, M. P. (2000). Web Surveys: A Review of Issues and Approaches. *The Public Opinion Quarterly, 64*, 464-494.

Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* Thousand Oaks: Sage Publications.

Descriptive Statistics. (n.d.). Retrieved June 20, 2015, from Research methods knowledge base: http://www.socialresearchmethods.net/kb/statdesc.php

Dingsøyr, T. D. (2010). *Agile Software Development: Current Research and Future Directions.* Berlin/Heidelberg: Springer.

Dingsøyr, T. N. (2012). A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems and Software, 85*, 1213– 1221.

Dybå, T. D. (2008). Empirical studies of agile software development: a systematic review. *Information and Software Technology, 50*, 833–859.

Dybå, T. D. (2009). What Do We Know about Agile Software Development? *IEEE Software*, 6-9.

Erickson, J. L. (2005). Agile Modeling, Agile software development, and extreme programming: the state of research. *Journal of Database Management, 16(4)*, 88–100.

Fogelström, N. T. (2010). The impact of agile principles on market-driven software product development. *Journal of Software Maintenance and Evolution: Research and Practice*, 53–80.

Fowler, M. H. (2001). The Agile Manifesto.

Glass, R. J. (2001). Agile versus Traditional: make love not war. *Cutter IT Journal*, 12-18.

Hastie, S. (2010, June 25). What does it mean to be Agile - survey results. Retrieved June 17, 2015, from http://www.infoq.com/news/2010/06/what-means-agile-survey

Highsmith, J. (2002). *Agile Software Development Ecosystems.* Boston: Addison-Wesley.

Highsmith, J. (2010). *Agile Project Managent: Creatinng Innovative Products.* Boston: Pearson Education.

History: The Agile Manifesto. (2001). Retrieved March 24, 2015, from Agile Manifesto: http://agilemanifesto.org/history.html

HP, L.P. (2015, May). Agile is the new normal.

Jalali, S. a. (2010). Agile practices in global software engineering-a systematic. *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference*, (pp. 45–54).

Kalermo, J. R. (2002). Agile Software development in theory and practice.

Kongyai, B. E. (2011). Adaptation of Agile Practices:A Systematic Review and Survey.

Larman, C. B. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer Society*, 47-56.

Livari, J. M. (1998). The usage of systems development methods: Are we stuck to old practices? *Information Software Technology*, 501–510.

Lyytinen, K. R. (2006). Information system development agility as organizational learning. *European Journal of Information Systems*, 183–199.

Martin, R. (2014). *Agile Software Development, Principles, Patterns, and Practices: Pearson New International Edition.* Essex: Pearson Education Limited.

Medinilla, A. (2012). *Agile Management Leadership in an Agile Environment.* Berlin Heidelberg: Springer.

Meyer, B. (2014). *Agile! The Good, the Hype and the Ugly .* Berlin: Springer.

Misra, S. K. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 1869-1890.

Muijs, D. (2004). *Doing Quantitative Research in Education with SPSS.* Sage Publications.

Nerur, S. M. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, 72-28.

Neuman, L. (2011). *Social Research Methods.*

Poppendieck, M. (2001). Lean Programming. *Software Development Magazine*, 71-75.

Rajlich, V. (2006). Changing the Paradigm of Software. *Communications of the ACM*, 67 - 70.

Rea, M. L. (2005). *Designing and Conducting Survey Research: A Comprehensive Guide.* San Fracisco: Jossey Bass.

Robson, C. (2002). *Real World Research.* Blackwell.

Salo, O. A. (2004). Empirical Evaluation of Agile Software Development: The Controlled Case Study Approach. In *Product Focused Software Process Improvement* (pp. 408-423). Berlin Heidelberg: Springer.

Schwaber, K. S. (2013, July). The Scrum Guide: Definitive guide to scrum – rules of the game. Retrieved April 15, 2015, from http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

Siau, K. (2005). A retrospective review of JDM from 2003 to 2005 and a discussion on publication emphasis of JDM for the next two to three years. *Journal of Database, 16*, 1.

Smith, G. S. (2009). *Becoming Agile in an imperfect world.*

Stapleton, J. (1997). *DSDM: Dynamic Systems Development Method.* Harlow, England: Addison-Wesley.

The Standish Group. (2013). Chaos Manifesto 2013: Think big, act small. Retrieved May 14, 2015, from http://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf

Turk, D. F. (2002). Limitations of Agile Software Processes. *Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*, (pp. 43-46). Alghero.

Turk, D. F. (2005). Assumptions Underlying Agile Software Development Processes. *Journal of Database Management*, 62-87.

VersionOne. (2015). State of Agile.

Vilain, P. M. (2011). Neglecting Agile Principles and Practices: A Case Study. *the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011).* Miami.

Vinekar, V. S. (2006). Can agile and traditional systems development approaches co-exist? An ambidextrous view. *Information Systems Management*, 31-42.

West, D. G. (2010). Agile Development: Mainstream Adoption Has Changed Agility. Retrieved March 25, 2015, from http://programmedevelopment.com/public/uploads/files/forrester_agile_development_mainstream_adoption_has_changed_agility.pdf

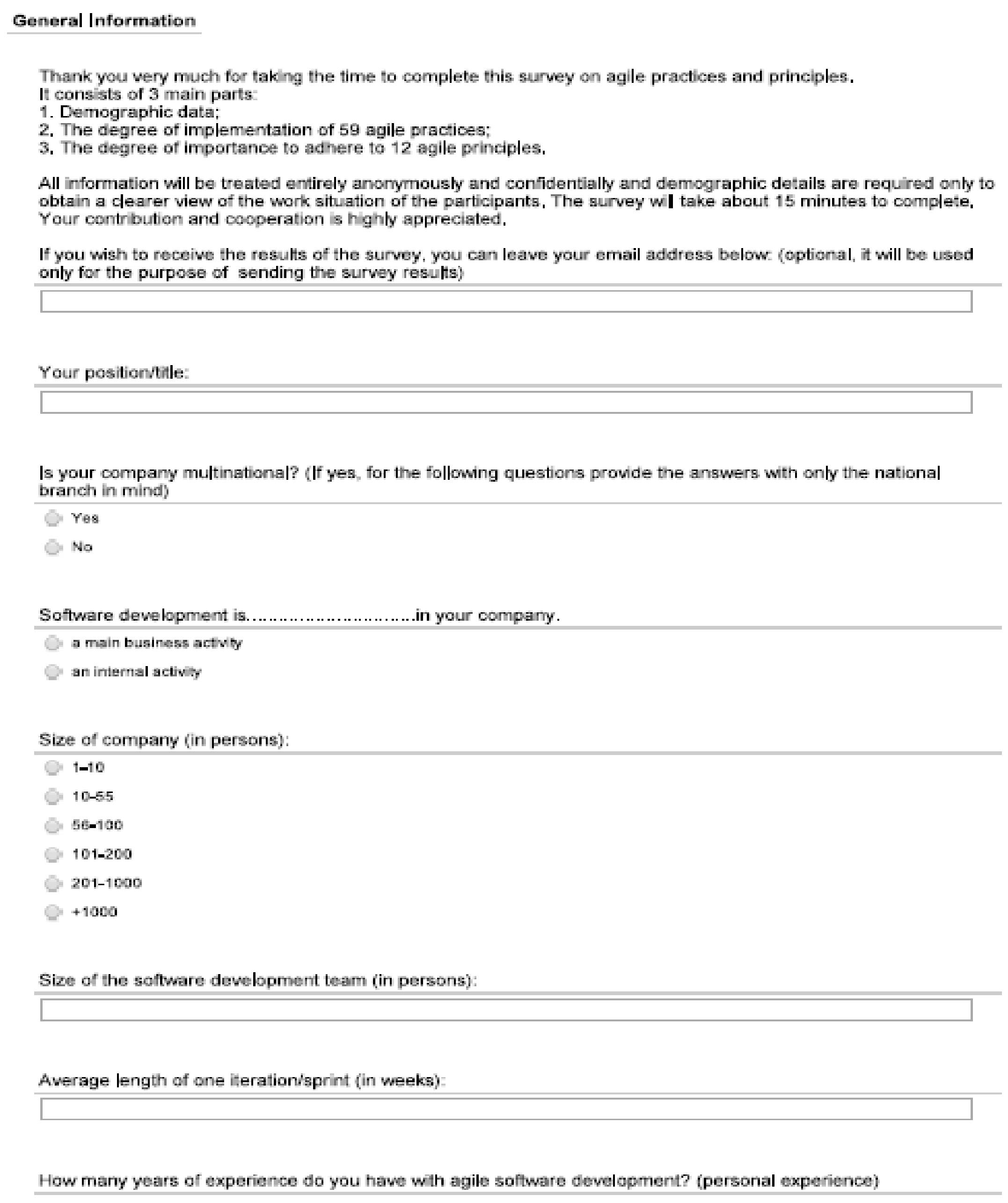
Williams, L. C. (2003). Agile Software Development: Its about feedback and change. *IEEE Computer Society*, 39-42.

Williams, L. D. (2014). Agile Software Development in Practice. In G. M. Cantone (Ed.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 32-45). Rome: Springer International Publishing.

Williams, L. R. (2010). Driving Process Improvement via Comparative Agility Assessment. *AGILE '10 Proceedings of the 2010 Agile Conference*, (pp. 3-10). Washington.

8. Appendix A: Online survey Questionnaire Design

Survey Questionnaire was designed and conducted using Qualtrics Survey Software tool.



General Information

Thank you very much for taking the time to complete this survey on agile practices and principles.
It consists of 3 main parts:
1. Demographic data;
2. The degree of implementation of 59 agile practices;
3. The degree of importance to adhere to 12 agile principles.

All information will be treated entirely anonymously and confidentially and demographic details are required only to obtain a clearer view of the work situation of the participants. The survey will take about 15 minutes to complete. Your contribution and cooperation is highly appreciated.

If you wish to receive the results of the survey, you can leave your email address below: (optional, it will be used only for the purpose of sending the survey results)

Your position/title:

Is your company multinational? (If yes, for the following questions provide the answers with only the national branch in mind)

- Yes
- No

Software development is..............................in your company.

- a main business activity
- an internal activity

Size of company (in persons):

- 1-10
- 10-55
- 56-100
- 101-200
- 201-1000
- +1000

Size of the software development team (in persons):

Average length of one iteration/sprint (in weeks):

How many years of experience do you have with agile software development? (personal experience)

Practices 1

Please specify your degree of implementation of the following Agile practices in your work process - PART 1/12 (list of 59 agile practices, for ease of use the list of practices has been divided into 12 parts) (N/A -not applicable) :

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully Implemented (>81%)	N/A
Acceptance tests/functional test (a formal description of the behavior of a software product, generally expressed as an example or a usage scenario)	○	○	○	○	○
ATDD (acceptance test driven development) this practice consists in the use of automated acceptance tests with the additional constraint that these tests be written in advance of implementing the corresponding functionality	○	○	○	○	○
Automated build The build is "automated" to the extent that these steps are repeatable and require no direct human intervention, and can be performed at any time with no information other than what is stored in the source code control repository.	○	○	○	○	○
Backlogs (Product and Sprint) a list of features or technical tasks which the team maintains and which, at a given moment, are known to be necessary and sufficient to complete a project or a release	○	○	○	○	○
Backlog grooming The team (or part of the team including the product owner) meet regularly to "groom the product backlog" (remove, add, reprioritise user stories, correct estimations), in a formal or informal meeting	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 2/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully Implemented (>81%)	N/A
Behavior-driven development (BDD) specifies that tests of any unit of software should be specified in terms of the desired behavior of the unit, it assumes the use of specialized software tools to support the development process	○	○	○	○	○
Burndown Chart a large graph relating the quantity of					

work remaining (on the vertical axis) and the time elapsed since the start of the project (on the horizontal, showing future as well as past)	○	○	○	○	○
Collective code ownership Collective code ownership, as the name suggests, is the explicit convention that "every" team member is not only allowed, but in fact has a positive duty, to make changes to "any" code file as necessary	○	○	○	○	○
Continuous integration minimize the duration and effort required by "each" integration episode, be able to deliver "at any moment" a product version suitable for release	○	○	○	○	○
CRC Cards (for Class, Responsibilities, Collaborators) a brainstorming tool used in the design of object-oriented software, the goal is to use role playing as a means to come up with a better design,	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 3/12

	Not implemented (<20%)	Partially implemented (20–50%)	Mostly implemented (51–80%)	Fully Implemented (>81%)	N/A
Continuous deployment an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by live users, in production.	○	○	○	○	○
Daily meeting (daily scrum or stand up) Each day at the same time, the team meets so as to bring everyone up to date on the information that is vital for coordination	○	○	○	○	○
Definition of done The team agrees on, and displays prominently somewhere in the team room, a list of criteria which must be met before a product increment "often a user story" is considered "done" and can be reviewed by product owner - for increment	○	○	○	○	○
Definition of ready is a set of rules or criteria that the team adopts as a guide for when a story can legitimately be moved from the backlog into a Sprint, Having a Definition of Ready means that stories must be immediately actionable, clear, concise , - for story	○	○	○	○	○

Estimation consists of a quantified evaluation of the effort necessary to carry out a given development task; this is most often expressed in terms of duration	○	○	○	○	○

Practices 2

Please specify your degree of implementation of the following Agile practices in your work process - PART 4/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Exploratory testing emphasizes the tester's autonomy, skill and creativity, recommends performing various test-related activities in an interleaved manner, throughout the project	○	○	○	○	○
Facilitation A facilitator is a person who chooses or is given the explicit role of conducting a meeting. This role usually entails that the facilitator will take little part in the discussions, but will focus primarily on creating the conditions for effective group processes (ex. ScrumMaster)	○	○	○	○	○
Frequent releases Agile team frequently releases its product into the hands of end users, listening to feedback, whether critical or appreciative (can be each iteration or in every 2-4 iteration)	○	○	○	○	○
Given-when-then - a template intended to guide the writing of acceptance tests for a User Story	○	○	○	○	○
Heartbeat retrospective (sprint/iteration retrospective) meetings to explicitly reflect on the most significant events to have occurred since the previous such meeting, and take decisions aiming at remediation or improvement,	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 5/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Information radiators generic term for any of a number of handwritten, drawn, printed or electronic displays which a team places in a highly visible location, so that all team members as well as passers-by can see the latest information at a glance	○	○	○	○	○
Integration ex: if two					

developers, working in parallel, implement new features on two components A and B, and each thinks to their own satisfaction that the work is complete, then verifying that changes to A and B are consistent, and resolving any inconsistencies, belong in the category of integration.	○	○	○	○	○
Invest acronym INVEST helps to remember a widely accepted set of criteria, or checklist, to assess the quality of a user story	○	○	○	○	○
Iteration/sprint a timebox during which development takes place	○	○	○	○	○
Iterative development Agile projects intentionally allow for "repeating" software development activities, and for potentially "revisiting" the same work products	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 6/12

	Not implemented (<20%)	Partially implemented (20–50%)	Mostly implemented (51–80%)	Fully Implemented (>81%)	N/A
Incremental development this means that each successive version of the product is usable, and each builds upon the previous version by adding user-visible functionality	○	○	○	○	○
Kanban board is not "reset" at the beginning of each iteration, its columns represent the different processing states of a "unit of value", which is generally equated with a user story	○	○	○	○	○
Milestone retrospective Once a project has been underway for some time, or at the end of the project (especially when the team is likely to work together again), all of the team's permanent members (not just the developers) invests from one to three days in a detailed analysis of the project's significant events	○	○	○	○	○
Mock objects used in the context of crafting automated unit tests and consists of instantiating a test-specific version of a software component (typically a class)	○	○	○	○	○
Niko Niko calendar (mood board) The format of the calendar allows each team member to record, at the end of every workday, a graphic evaluation (with emoticon or coloured sticker) of their mood during that day.	○	○	○	○	○

Lead time/cycle time is measurement practice and means time-space between the formulation of a user story and that story being used "in production" as a ready software functionality	○	○	○	○	○

Practices 3

Please specify your degree of implementation of the following Agile practices in your work process - PART 7/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Pair programming two programmers sharing a single workstation (driver and navigator)	○	○	○	○	○
Personas are used to describe stakeholders/users of system, it is concise and visual; a common layout is a single page including a photograph, a name and social or professional details: "Amanda Jones, 34, press officer at a major food retailing organization, etc."	○	○	○	○	○
Points (estimate in) most widespread unit for estimate is "story points" (also gummy bears)	○	○	○	○	○
Planning poker A playful approach to estimation, The team meets in presence of the customer/Product Owner, Around the table, each team member holds a set of playing cards, bearing numerical values appropriate for points estimation of a user story.	○	○	○	○	○
Project chartering The team develops and maintains a high-level summary of the project's key success factors, synthetic enough that it can be displayed on one wall of the team room as a flipchart-sized sheet of paper	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 8/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Quick design session when there is a need for further design, two or more developers meet for a quick design session at the whiteboard, possibly using design aids such as CRC cards.	○	○	○	○	○
Refactoring consists of improving the internal structure of an existing program's source code, while preserving its external behaviour.	○	○	○	○	○

Relative estimation consists of estimating tasks or user stories, not separately and in absolute units of time, but by comparison or by grouping of items of equivalent difficulty.	○	○	○	○	○
Role-feature reason - most commonly recommended aids for teams and product owners starting to write user stories: As a...I want...So that...	○	○	○	○	○
Rules of simplicity a set of criteria, in priority order, to judge whether some source code is "simple enough"	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 9/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Scrum of scrums a technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10.	○	○	○	○	○
Simple design/ YAGNI, for "You Aren't Gonna Need It", emergent ongoing design, as simple as possible	○	○	○	○	○
Sign up for tasks members of an Agile development team normally choose which tasks to work on, rather than being assigned work by a manager	○	○	○	○	○
Story splitting consists of breaking up one user story into smaller ones, while preserving the property that each user story separately has measurable business value	○	○	○	○	○
Story mapping consists of ordering user stories along two independent dimensions. The "map" arranges user activities along the horizontal axis in rough order of priority. Down the vertical axis, it represents increasing sophistication of the implementation.	○	○	○	○	○

Practices 4

Please specify your degree of implementation of the following Agile practices in your work process - PART 10/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Sustainable pace entails a firm refusal of what is often considered a "necessary evil" in the software industry -	○	○	○	○	○

long work hours, overtime, or even working nights or weekends.					
Task board can be drawn on a whiteboard or even a section of wall, the board is divided into three columns labelled "To Do", "In Progress" and "Done", reflecting the current status of the tasks, updated frequently, most commonly during the daily meeting, reset" at the beginning of each iteration	○	○	○	○	○
Test-driven development (TDD) refers to a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design (in the form of refactoring), first write test and then code	○	○	○	○	○
Team (whole) a small group of people (including product owner), assigned to the same project or effort, nearly all of them on a full-time basis, Shared accountability, cross functional	○	○	○	○	○
Team room dedicated space for team for the duration of the project, set apart from other groups' activities, Workstations, whiteboards, enough wall space for task boards and so on.	○	○	○	○	○

Please specify your degree of implementation of the following Agile practices in your work process - PART 11/12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully Implemented (>81%)	N/A
Three C's "Card, Conversation, Confirmation" components of user story	○	○	○	○	○
Time-box a previously agreed period of time during which a person or a team works steadily towards completion of some goal. The critical rule of timeboxed work is that work should stop at the end of the timebox, and review progress	○	○	○	○	○
User stories in consultation with the customer or product owner, the team divides up the work to be done into functional increments called "user stories"	○	○	○	○	○
Three questions The daily meeting is structured around some variant of the following three questions: What have you completed since the last meeting? What do you plan to complete by the next meeting? What is getting in your way? The intent of	○	○	○	○	○

these questions is to emphasize completions of tasks, rather than effort spent.					
Usability testing a long-established, empirical and exploratory technique to answer questions such as "how would an end user respond to our software under realistic conditions?"	○	○	○	○	○

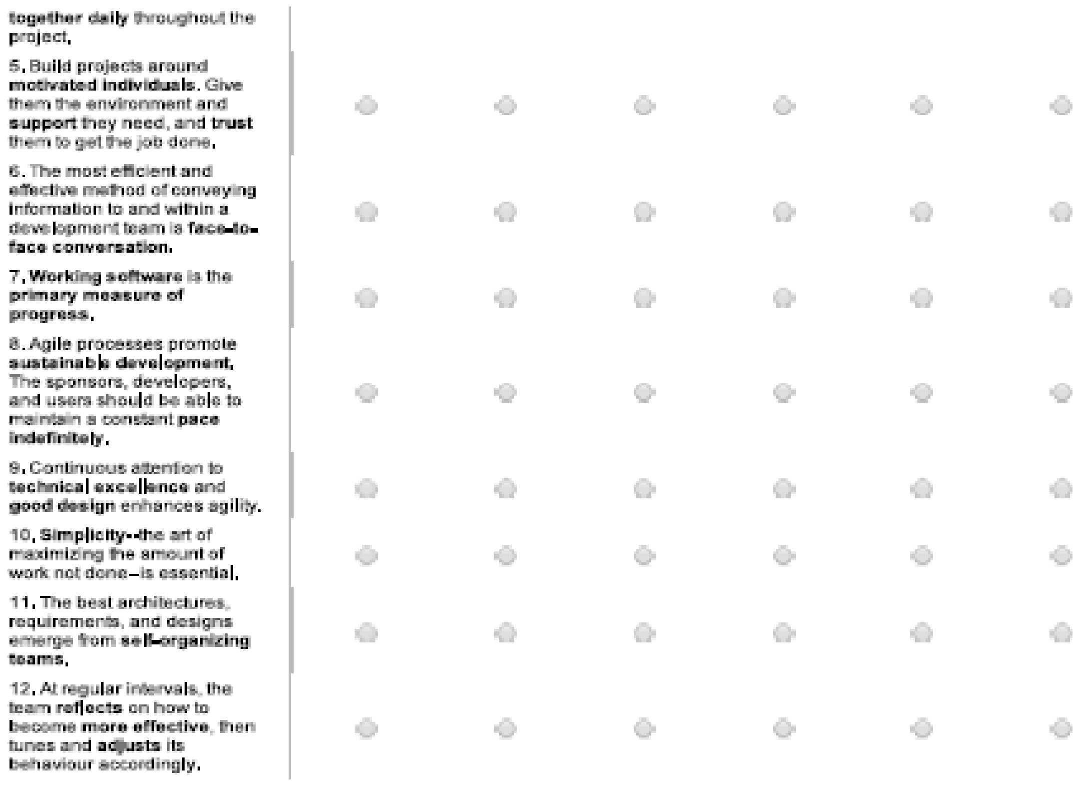
Please specify your degree of implementation of the following Agile practices in your work process - PART 12

	Not implemented (<20%)	Partially implemented (20-50%)	Mostly implemented (51-80%)	Fully implemented (>81%)	N/A
Ubiquitous language consists notably of striving to use the vocabulary of a given business domain, not only in discussions about the requirements for a software product, but in discussions of design as well and all the way into "the product's source code itself".	○	○	○	○	○
Unit testing a short program fragment written and maintained by the developers on the product team, which exercises some narrow part of the product's source code and checks the results.	○	○	○	○	○
Velocity at the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity.	○	○	○	○	○
Version control enabler of a number of Agile practices, such as continuous integration	○	○	○	○	○

Principles

Below is a list of 12 Agile principles. Please evaluate them according to their importance for your agile software development process: (Last Question! Almost Done)

	Unimportant	Partly unimportant	Neither unimportant, nor important	Partly important	Important	Very important
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 behaviour accordingly.	○	○	○	○	○	○

9. Appendix B: Mapping design

Practices/Principles mapping	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 behaviour accordingly.
Acceptance tests/functional test a formal description of the behaviour of a software product, generally expressed as an example or a usage scenario												
ATDD (acceptance test driven development) this practice consists in the use of automated acceptance tests with the additional constraint that these												

tests be written in advance of implementing the corresponding functionality												
Automated build The build is "automated" to the extent that these steps are repeatable and require no direct human intervention, and can be performed at any time with no information other than what is stored in the source code control repository.												
Backlogs (Product and Sprint) a list of features or technical tasks which the team maintains and which, at a given moment, are known to be necessary and sufficient to complete a project or a release												
Backlog grooming The team (or part of the team including the product owner) meet regularly to "groom the product backlog" (remove, add, reprioritise user stories, correct estimations), in a formal or informal meeting												
Behaviour-driven development (BDD) specifies that tests of any unit of software should be specified in terms of the desired behaviour of the unit,												

it assumes the use of specialized software tools to support the development process												
Burndown Chart a large graph relating the quantity of work remaining (on the vertical axis) and the time elapsed since the start of the project (on the horizontal, showing future as well as past)												
Collective code ownership - as the name suggests, is the explicit convention that "every" team member is not only allowed, but in fact has a positive duty, to make changes to "any" code file as necessary												
Continuous integration minimize the duration and effort required by "each" integration episode, be able to deliver "at any moment" a product version suitable for release												
CRC Cards (for Class, Responsibilities, Collaborators) a brainstorming tool used in the design of object-oriented software, the goal is to use role playing as a means to come up with a better design.												

Continuous deployment an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by live users, in production.												
Daily meeting (daily scrum or stand up) Each day at the same time, the team meets so as to bring everyone up to date on the information that is vital for coordination												
Definition of done The team agrees on, and displays prominently somewhere in the team room, a list of criteria which must be met before a product increment "often a user story" is considered "done" and can be reviewed by product owner - increment driven												
Definition of ready is a set of rules or criteria that the team adopts as a guide for when a story can legitimately be moved from the backlog into a Sprint, Having a Definition of Ready means that stories must be immediately actionable, clear,												

concise . -story driven												
Estimation consists of a quantified evaluation of the effort necessary to carry out a given development task; this is most often expressed in terms of duration												
Exploratory testing emphasizes the tester's autonomy, skill and creativity, recommends performing various test-related activities in an interleaved manner, throughout the project												
Facilitation A facilitator is a person who chooses or is given the explicit role of conducting a meeting. This role usually entails that the facilitator will take little part in the discussions, but will focus primarily on creating the conditions for effective group processes (ex. ScrumMaster)												
Frequent releases Agile team frequently releases its product into the hands of end users, listening to feedback, whether critical or appreciative (can be each iteration or in every 2-4												

iteration)												
Given-when-then a template intended to guide the writing of acceptance tests for a User Story												
Heartbeat retrospective (sprint/iteration retrospective) meetings to explicitly reflect on the most significant events to have occurred since the previous such meeting, and take decisions aiming at remediation or improvement.												
Information radiators generic term for any of a number of handwritten, drawn, printed or electronic displays which a team places in a highly visible location, so that all team members as well as passers-by can see the latest information at a glance												
Integration ex: if two developers, working in parallel, implement new features on two components A and B, and each thinks to their own satisfaction that the work is complete, then verifying that changes to A and B are consistent, and												

resolving any inconsistencies, belong in the category of integration.												
Invest acronym INVEST helps to remember a widely accepted set of criteria, or checklist, to assess the quality of a user story												
Iteration/sprint a timebox during which development takes place												
Iterative development Agile projects intentionally allow for "repeating" software development activities, and for potentially "revisiting" the same work products												
Incremental development this means that each successive version of the product is usable, and each builds upon the previous version by adding user-visible functionality												
Kanban board is not "reset" at the beginning of each iteration, its columns represent the different processing states of a "unit of value", which is generally equated with a user story												
Lead time/cycle time is measurement practice and means time-space between the formulation of a												

user story and that story being used "in production" as a ready software functionality												
Milestone retrospective Once a project has been underway for some time, or at the end of the project (especially when the team is likely to work together again), all of the team's permanent members (not just the developers) invests from one to three days in a detailed analysis of the project's significant events												
Mock objects used in the context of crafting automated unit tests and consists of instantiating a test-specific version of a software component (typically a class)												
Niko Niko calendar (mood board) The format of the calendar allows each team member to record, at the end of every workday, a graphic evaluation (with emoticon or coloured sticker) of their mood during that day.												
Pair programming two programmers sharing a single workstation (driver and navigator)												

Personas are used to describe stakeholders/users of system, it is concise and visual; a common layout is a single page including a photograph, a name and social or professional details: "Amanda Jones, 34, press officer at a major food retailing organization, etc."												
Points (estimate in) most widespread unit for estimate is "story points" (also gummy bears)												
Planning poker A playful approach to estimation. The team meets in presence of the customer/Product Owner. Around the table, each team member holds a set of playing cards, bearing numerical values appropriate for points estimation of a user story.												
Project chartering The team develops and maintains a high-level summary of the project's key success factors, synthetic enough that it can be displayed on one wall of the team room as a flipchart-sized sheet of paper												

Quick design session when there is a need for further design, two or more developers meet for a quick design session at the whiteboard, possibly using design aids such as CRC cards.												
Refactoring consists of improving the internal structure of an existing program's source code, while preserving its external behaviour.												
Relative estimation consists of estimating tasks or user stories, not separately and in absolute units of time, but by comparison or by grouping of items of equivalent difficulty.												
Role-feature reason - most commonly recommended aids for teams and product owners starting to write user stories: As a...I want...So that...												
Rules of simplicity A set of criteria, in priority order, to judge whether some source code is "simple enough"												
Scrum of scrums A technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10.												

Simple design YAGNI, for "You Aren't Gonna Need It", emergent ongoing design, as simple as possible												
Sign up for tasks Members of an Agile development team normally choose which tasks to work on, rather than being assigned work by a manager												
Story splitting consists of breaking up one user story into smaller ones, while preserving the property that each user story separately has measurable business value												
Story mapping consists of ordering user stories along two independent dimensions. The "map" arranges user activities along the horizontal axis in rough order of priority. Down the vertical axis, it represents increasing sophistication of the implementation.												
Sustainable pace entails a firm refusal of what is often considered a "necessary evil" in the software industry - long work hours, overtime, or even working nights or weekends.												

Task board can be drawn on a whiteboard or even a section of wall. The board is divided into three columns labelled "To Do", "In Progress" and "Done", reflecting the current status of the tasks. updated frequently, most commonly during the daily meeting, reset" at the beginning of each iteration												
Test-driven development (TDD) refers to a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design (in the form of refactoring), first write test and then code												
Team (whole) a small group of people (including product owner), assigned to the same project or effort, nearly all of them on a full-time basis. Shared accountability, cross functional												
Team room dedicated space for team for the duration of the project, set apart from other groups' activities. Workstations, whiteboards, enough wall space for task boards and so on.												

Three C's "Card, Conversation, Confirmation" components of user story												
Three questions The daily meeting is structured around some variant of the following three questions: What have you completed since the last meeting? What do you plan to complete by the next meeting? What is getting in your way? The intent of these questions is to emphasize completions of tasks, rather than effort spent.												
Time-box a previously agreed period of time during which a person or a team works steadily towards completion of some goal. The critical rule of timeboxed work is that work should stop at the end of the timebox, and review progress												
User stories In consultation with the customer or product owner, the team divides up the work to be done into functional increments called "user stories"												
Usability testing a long-established, empirical and exploratory technique												

to answer questions such as "how would an end user respond to our software under realistic conditions?"												
Ubiquitous language consists notably of striving to use the vocabulary of a given business domain, not only in discussions about the requirements for a software product, but in discussions of design as well and all the way into "the product's source code itself".												
Unit testing a short program fragment written and maintained by the developers on the product team, which exercises some narrow part of the product's source code and checks the results.												
Velocity At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity.												
Version control enabler of a number of Agile practices, such as continuous integration												

10. Appendix C: Invitation letters for participation in online survey

Dear Agile practitioners,

I am working on my Master's Thesis which is about the importance and the presence of Agile principles in Agile software development. The goal of the survey is to identify the degree of implementation of agile practices and perceived importance of agile principles.

Could you please take a few minutes to share with me your valuable experience on your degree of implementation of Agile practices and on the importance of agile principles for your development process by completing this anonymous survey?

Your input as an agile practitioner is critically important for my research and I highly appreciate your contribution to my master thesis. You will be provided with the survey results afterwards if you wish so.

Below is the link to the survey. Thank you in advance for your time and input. Please feel free to share this survey link with other agile practitioners.

https://ugenthabe.az1.qualtrics.com/SE/?SID=SV_4T5B084UVbB2rs1

Kind regards,

Aygun Shafagatova

11. Appendix D: Invitation Letters for participation in mapping study

Dear,

I am doing my Master in Management and IT at the University of Ghent, Belgium. I am currently researching on agile principles for my Master Thesis. 'The presence and importance of agile principles in Agile software development' is my research topic. As you have deep expertise/knowledge on agile methodologies, it would be very valuable for my research to know your vision and analysis regarding those issues.

I need your feedback to be able to make possible correlation between Agile principles and practices. For this end, I have prepared an Excel mapping document which contains a matrix of 12 agile principles and 60 practices. I would like you to take it through and match/map those practices with principles. According to your experience and vision: which practice is based on or supports which principle? Just put 'x' where you consider it is relevant. There may be cases that no matches can be made, or multiple practices can match same principle (or multiple principles for the same practice), that is also perfectly OK. For the ease of use, I have divided the mapping matrix into 6 sheets so that you don't have to scroll up and down (such as Agile Mapping 1 of 6, etc).

Please find attached Excel mapping document. May I kindly ask you to make the mapping and send it back to me after completing it? It will not take much time of you; however, it will deliver highly valuable input for my thesis.

Thank you in advance for your time and cooperation,

Looking forward to hearing from you soon,

Kind regards,

Aygun Shafagatova