Universiteit Leiden
Master ICT in Business

Agile Maturity and Quality Metrics

Name: Mohsen Rezai

Student-no: s1330446

Date: July 2015

1st supervisor: Dr. Werner Heijstek

2nd supervisor: Dr. Christoph Johann Stettina

Master Thesis

Leiden Institute of Advanced Computer Science
(LIACS) Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ACKNOWLEDGMENTS

First of all I want to express my gratitude to everyone involved in supporting this thesis research project. I would like to thank the management of involved organizations in Netherlands, United Kingdom and Israel for their support and brining me in touch with the relevant people in their organization. The individuals of the representing organizations have really contributed to shape this study, I am thankful for your effort.

I would like to thank my first supervisor, Werner Heijstek for his support, guidance and helping me through the chosen path of this thesis. And finally, I would like to thank my second supervisor, Christoph Johann Stettina for his feedback and support.

ABSTRACT

Implementing agile methodologies according to the principles (Agile manifesto, 2001) can be challenging. Agile maturity and ensuring high software quality are imperative aspects to software development organizations. Agile development promises to provide quality assurance (Sirshar & Arif, 2012), but how is agile maturity related to software quality metrics? We conducted a multiple case study in three countries across 11 organizations to answer this question. With aid of semi-structured interviews and surveys, we collected the necessary data to support our research. We measured the teams on agility aiming on multiple dimensions including software quality, testing process, agile practices, software deliveries and organizational strategy. Using this, we analyzed what quality metrics are implemented and how they affect agile maturity.

We concluded that the relation between agile maturity and quality metrics is most apparent in low and high mature organizations, considering that the use of quality metrics is not very popular. The focus of low mature organizations is mainly to improve on existing agile practices and the use of quality metrics is neglected. Organizations with higher maturity are focused to increase quality, using quality metrics related to testing aspects. As a result, agile maturity leads to higher software quality, indicated by effective implementation of agile practices related to software quality.

Table of Content

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The broad introduction of agile methods such as XP and scrum (Dyba & Dingsøyr, 2008) has triggered many organizations to implement these methods and benefit from it. It's been a few years that agile methodologies have won territory from the waterfall method. This is mostly related to the fact that agile methods can be implemented fast, and is flexible due to frequent feedback loops, iterative reviews and close contact with the business (Stettina & Hörz, 2015).

The most popular agile method is scrum, with a majority of 73% (State of agile survey, 2014). Scrum aims to replace command-and control management with collaborative self-managing teams (Moe, Dingsøyr, & Røyrvik, 2009). The team autonomy is an important subject within scrum.

1.1 Motivation

Within agile and specifically scrum, is empowering teams an important factor. Agile implementation and self-organizing teams can be helped by increased development team self-awareness (Stettina & Heijstek, 2011). This is very much related to growth of team in agile development. But how is this growth determined and what factors are crucial for agile growth? And the question remains how this growth affects software quality.

Measuring organization's agility based on the teams seems to be a challenge. Organizations want to discover what the improvement areas are to grow in agile development. As a result, there are many agile maturity models (Ozcan-Top & Demirörs, 2013) that provide insight regarding how to measure maturity. It is interesting to discover what the relation is between agile maturity and software quality. Organizations that have been implementing agile for a couple of years can be considered becoming mature. But how will this maturity affect the software quality? Particularly, what is the effect of agile maturity on software quality metrics? These are the interesting questions that we want to answer with this research.

From the quality perspective, agile development promises to provide quality assurance (Sirshar & Arif, 2012). But on the other hand, these methodologies do not explicitly provide practices for managing and measuring quality and reliability, as described in ISO/IEC 9126 (Jinzenji, Williams, Hoshino, & Takahashi, 2013). In this study we want to discover how this quality assurance is realized with use of particular quality metrics. And are these agile or traditional metrics? Approaches from traditional Quality Assurance are independent from the underlying Development Methodology, and if combined well, agile software development may benefit from these (Janus, Schmietendorf, Dumke, & Jäger, 2012). This study will discover the applied quality metrics in organizations, and will spot any relation to agile maturity of organizations.

1.2 Research subject

The use of Agile methods have increasingly attracted interest in the current software industry environment (Cardozo & Neto, 2009). Lack of experience in agile methods is one of the leading causes that agile projects fail (State of agile survey, 2014).

Agile maturity can be measured using models and frameworks (Packlick, 2007; Patel & Ramachandran, 2009; Qumer & Henderson-Sellers, 2009; Sidky & Arthur, 2007; Yin & Figueiredo, 2011; Soares & Meira, 2013). See below for a list of few popular selected models and frameworks.

Model/framework	Authors	Number of levels
Agile maturity model (AMM)	Patel & Ramachandran (2009)	5
Agile maturity map (AMM)	Packlick (2007)	5
Agile Adoption and Improvement Model (AAIM)	Qumer & Henderson-Sellers (2008)	6
Sidky Agile Measurement Index (SAMI)	Sidky (2007)	5
Scrum maturity model (SMM)	Yin & Figueiredo (2011)	5

Table 1: Agile maturity models and frameworks

In all described models there is a consensus that achieving higher maturity results in higher quality software. In addition, metrics play an important role in establishing maturity: The higher levels of maturity require many metrics - depending on the model used.

More mature agile development practices are expected to result in higher quality software. Software quality metrics are therefore expected to play a role in determining agile maturity. However, there is no clear indication what the role of software quality metrics is in achieving a higher maturity.

Consequently, we propose the following (main) research question:

How are the maturity of an agile software development approach and the use of software quality metrics related?

Research sub-questions:

- RQ1: What are the most effective quality metrics that are being used in agile organizations?
- RQ2: What are the most effective agile practices that support maturity?
- RQ3: How do organizations measure the maturity of their agile software development?
- RQ4: What is the impact of successful or failed projects on quality metrics?
- RQ5: At what maturity levels are which software quality metrics implemented?

Agile software development differs from traditional software development. Enhancing quality is a reason for organizations to adopt agile, and according to agile survey, 66% of the organizations have prioritized this as very important (State of agile survey, 2014). Quality plays an essential role in the fast pace software development environment (Imreh & Raisinghani, 2011). It is interesting to find out how agile organizations cope with the fast pace development and ensuring high quality and its relation to maturity. The ISO/IEC 9126 standard intends to ensure the quality of all software products and it consists of four parts: quality model, external metrics, internal metrics and quality in use metrics (ISO/IEC 9126, 2001).

See below for few possible quality metrics that can be used in agile organizations and are relevant to this research. This list will not represent the actual selected quality metrics, it is purely stated to provide an example which can be used in this research.

- Customer satisfaction (external quality)
- Code complexity (internal quality)
- Reported defects (external quality)
- Coverage (internal quality)
- Level of refactoring (internal quality)
- Lines of code (internal quality)

There are many quality metrics that organizations can apply to measure quality. An example can be customer satisfaction (external quality), this can be measured by post-release quality, it includes the number of defects delivered to and reported by the customer (Layman, Williams, & Cunningham, 2006; Sfetsos & Stamelos, 2010). A study on quality metrics showed that implementing agile software development and measuring post-release quality can contribute to a similar or better productivity than the industry average (Layman, Williams, & Cunningham, 2006).

Another quality metric can be refactoring, that leads to a higher code reuse and better quality (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008; Kunz, Dumke, & Schmietendorf, 2008). However, (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008) discusses that “the majority of software developers and researchers agree that refactoring has long-term benefits of the quality of a software product (in particular on program understanding) there is no such consensus regarding the development productivity” (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008). All the aforementioned software quality metrics are used in agile development methods and can be related to agile maturity.

1.3 Research scope

This study will only be focused on organizations that are implementing agile methodologies. Any agile method can be part of this study, there is no selection made between the popular methods. Due to the increased interest of organizations in agile methodologies, this study will only focus on the agile maturity of the organizations and the use of particular quality metrics in agile teams.

This study will be based on a multiple case study in 11 organizations in total. We have selected organizations that just started implementing agile, and organizations that have been implementing agile methodologies for seven years already. This range should provide a low, medium and high level maturity.

1.4 Relevance

This research has relevance to science and the practical use of agile methods with regards to agile maturity and software quality metrics. On the scientific base, there are no significant studies conducted on the relation of these aspects. The aim of this study is to contribute to the practical use of software quality metrics in agile projects and provide ways to measure agile maturity within organizations. In addition, this study aims to provide a general overview of the agile maturity levels of organizations and the corresponding imperative aspects of quality related to that specific maturity level.

1.5 Thesis outline

The structure of this study consists of six chapters. In chapter 1, *Introduction*, we introduce the topic, research subject, and the research questions.

In chapter 2, *Theoretical Framework*, the related work is reviewed based on the topics of agile maturity models and software quality metrics. This chapter provides the necessary knowledge and base for this study.

In chapter 3, *Methods,* all the methods applied in this research are described. This includes research design, data collection, and case selection strategy methods. Furthermore we describe the data analysis approach.

In chapter 4, *Results,* the findings of this study will be described.

In chapter 5, *Discussion*, we discuss the results by further elaborating these and trying to find out what these means.

In chapter 6, *Conclusion*, we conclude the findings and the main research contributions are outlined. Furthermore, the future research will be described.

2 THEORETICAL FRAMEWORK

In the following sections the literature based on the related work is reviewed. First we will look into maturity and study agile maturity models in order to understand how agile maturity is measured and how it aids this study. A summary of each agile maturity model is provided in section 2.3. Subsequently, in section 2.4 we will focus on quality, and in section 2.5 we will specifically review literature related to quality metrics.

2.1 Agile vs waterfall method

Software development has undergone a transformation in the past years. Agile methodologies have found their place in most of the software development practices and are replacing popularity of traditional software development methods such as the waterfall method. Based on a survey conducted by (Begel & Nagappan, 2007), agile methodology is popular due to rapid releases, flexibility of design and improved communication between team members.

Agile methodologies have gained popularity in the past years (Yin & Figueiredo, 2011) and many organizations are moving towards this emerging method. Agile methodologies have been around for some years, and organizations are trying to react in the fast pace environment of IT. IT organizations are required to deal with the fast technological changes and adapt to the extremely fluid environment, become more efficient and responsive in relationship to continuously rapidly changing environment, ensuring continuous future growth and prosperity (Kassim & Zain, 2004).

2.2 Maturity

Maturity in software development can be measured by implementing Capability Maturity Models (CMM) or Capability Maturity Model Integrations (CMMI). CMM is developed by Software Engineering Institute (SEI) and is probably the best known model to improve software processes (Paulk, 1999). CMM consists of five levels to improve the maturity of software processes (Paulk, Konrad, & Garcia, CMM Versus SPICE Architectures , 1995). The five levels are defined as:

1. Initial:
   The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2. Repeatable:
   Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3. Defined:
   The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4. Managed:
   Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5. Optimizing:
   Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Capability Maturity Model Integration (CMMI) is developed by the Software Engineering Institute (SEI) in 2006 to integrate and standardize the separate models of CMM. CMMI models are

implemented by organizations to improve development, acquisition and maintenance processes (Alegria & Bastarrica, 2006). CMMI models are generally used to establish improvement objectives, improving the processes, offering guidelines to create stable and mature processes. Like the agile maturity models, there are many CMMI models available that are designed for specific organizations. Based on the organizational structure, a proper CMMI model can be identified and implemented. The processes of CMMI models are staged in five maturity levels to enable support and improvements.

Standardized Process Improvement for Construction Enterprises (SPICE) is a process improvement framework that is implemented in the construction industry (M.Sarshar, et al., 2000). SPICE is an integration of CMM quality models and ISO/IEC 15504 (Alegria & Bastarrica, 2006). SPICE has been identifying the benefits of CMM to develop a construction specific framework.

These models are hard to implement in an agile environment, as a result, agile maturity models have been introduced. In section 2.4 we review agile maturity models.

2.3 Literature - Agile maturity models

This section focuses on agile maturity models specifically mentioned in the literature. According to (Kohlegger, Maier, & Thalmann, 2009), maturity models are instruments used to rate capabilities, and based on this rating, initiatives can be implemented to improve the maturity of an element—a person, an object or a social system. We have been studying the most popular maturity models and listed these in the following sections. Initially we provide an overview of each agile maturity model, then every model is reviewed in detail.

2.3.1 Overview of Agile maturity models

Table 1 provides an overview which summarizes the reviewed literature and presents an overview of all maturity models and their relation to specific characteristics. We analyzed how many levels a model provides, and does the level indicate what the characteristics of this levels are. The core characteristics should provide the overview of the level in order to differentiate between the levels. We analyzed whether the model is derived from the Capability Maturity Model Integration (CMMI). The level objective should provide the necessary goals to achieve in order to rise to the next level. Furthermore, we analyzed on which agile method the model is applicable. And finally, we looked if the model provides any naming convention for the levels.

Maturity model characteristics

Model	#levels	Detailed level characteristics	CMMI related	Level objectives	Agile method	Definition of levels	Levels naming
Agile adoption and improvement model (AAIM)	6	Yes	No	Yes	All	Yes	Yes
Agile maturity model (AMM)	5	Yes	No	Yes	All	Yes	Yes
Agile maturity map (AMM)	5	No	No	No	All	Yes	Yes
Maturity model for SD organizations	5	No	Yes	No	All	Yes	No
Agile adoption framework (AAF)	5	Yes	No	Yes	All	Yes	Yes
Scrum maturity model (SMM)	5	Yes	No	Yes	Scrum	Yes	Yes

Table 2: Overview of maturity models with corresponding characteristics

In table 3 we present the summary of all maturity models with their content related to specific levels. We analyzed each model and their corresponding level and described what the levels are consisted of. This table shows the differences and similarities in maturity levels, according to each level.

Level focus points

Model	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Agile adoption and improvement model (AAIM)	Speed, flexibility, responsiveness	Communication	Artifacts, documentation	People-oriented	Learning	Lean production, keep agile
Agile maturity model (AMM)	Process improvement, goals	Communication	Customer relationship, development practices and quality	People orientation, project management practices	Performance management, defect prevention	

Agile maturity map (AMM)	Goals and values	Development practices	Agile practices	Continuous improvement and innovation	Coaching, mentoring, learning	
Maturity model for SD organizations	Initial, no methodology used	Planning and monitoring projects	Standard processes	Manage processes with agile metrics	Continuous improvement and lean SW development	
Agile adoption framework (AAF)	Communication and collaboration	Early and continuous delivery	High quality and continuous improvement	Documentation, customer relationship	Sustain agility	
Scrum maturity model (SMM)	Process improvement, goals	Agile practices, quality satisfaction	Customer relationship, iteration management	Standard processes, process performance management	Performance management	

Table 3: Specific details of maturity levels

2.3.2 Agile software solution framework (ASSF)

Agile software solution framework (ASSF) is developed by (Qumer & Henderson-Sellers, 2009) in order to assist managers in assessing the degree of agility they require and how to identify appropriate ways to introduce this agility into their organization. This model contains three conceptual aspects: knowledge, governance and method core.

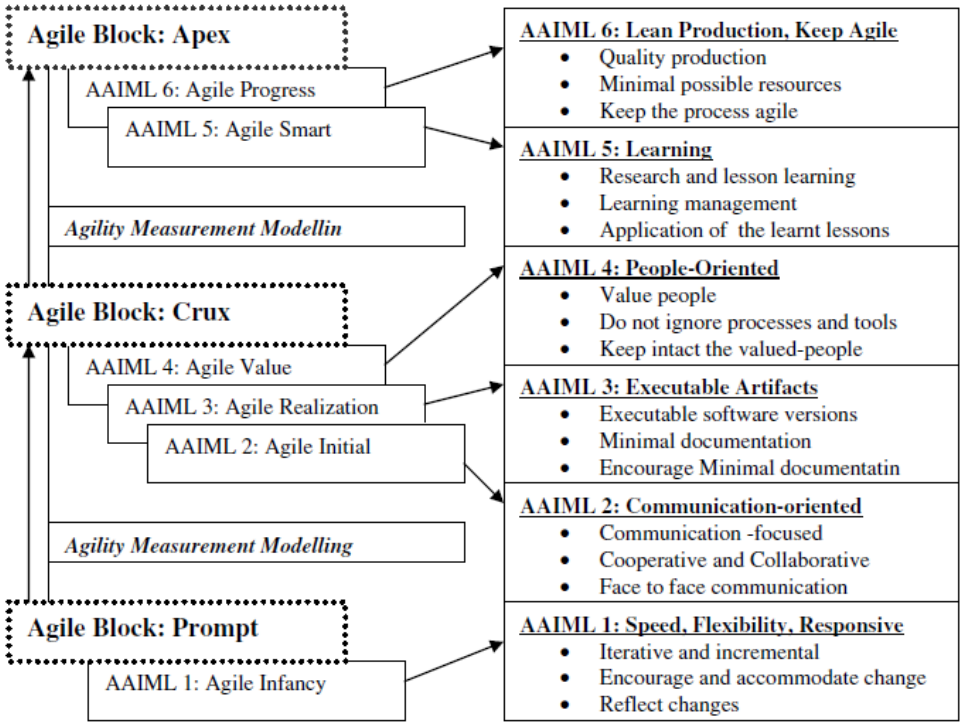

Figure 1: AAIM by Qumer and Henderson-Sellers

Furthermore, this model (Qumer & Henderson-Sellers, 2009) is developed to assess agility, adoption and improvements of agile processes. This model is called the Agile Adoption and Improvement Model (AAIM) and is method independent. The degree of agility of an agile process is measured quantitatively by using the agility measurement modelling approach (the 4-DAT tool) which is a

toolkit created to measure level of agility. This model consists of three blocks. There are six levels available, spread over three blocks; prompt, crux and apex. These blocks represent the agility from basic to advance.

The prompt block consists of a single level AAIM 1, called agile infancy. At this level the software development organizations don't implement standard agile methods, they rather apply basic parts of agile properties (speed, flexibility and responsiveness). Release and iteration planning are part of this level. The major goal at this level is to remain flexible and apply responsiveness to changes.
The second block is called crux and contains of three levels; agile initial, agile realization and agile value. At AAIM 2 level, which is the agile initial, the focus point is to create an internal communication pipelines between all the stakeholders. This means a good communication line between the co-workers, teams, and within the organizations itself. Establishment of external communication is required, which means communication with the customers and relevant external stakeholders.
Level 3 (AAIM 3) is the agile realization. This level aims to reduce the amount of documentation during the production of artefacts. The belief is that; if there is a good communication form e.g. face to face, verbal or other types of it, the need for documenting materials will be minimized. AAIM 4 focuses on agile values. At this point, the practices are established and the focus has also been pinpointed at people. Both, people within the organization and people outside of the organization (customers) are valued without ignoring the software development tools and processes. It is notable to mention that AAIM 1, 2 and 3 create the platform to achieve AAIM 4.
The last block is the apex, this block contains two levels; agile smart and agile process. This block focusses on learning and quality aspects. At this point, the stress on quality will be increased with a focus on minimal use of resources with continuous improvement of the agile environment. AAIM 5 is the fifth level and is called agile smart. This level focuses to establish a learning environment or the organization, teams and products. The learning engages all the stakeholders in the software development process (before, during and after the execution of a software process). Use of new tools and technology are part of this level and all together and will improve the organization. AAIM 6 focusses on creating a lean production environment with high quality and minimal use of resources and time frame.

2.3.3 Agile maturity model

The AMM is a model that has been created by Patel and Ramachandran and is a generic process that focusses on agile software development values, principles and practices (Patel & Ramachandran, 2009). It examines the agile practices conducted within an organization and links this to maturity levels. However, it's not a complete representation of agile software development practices.

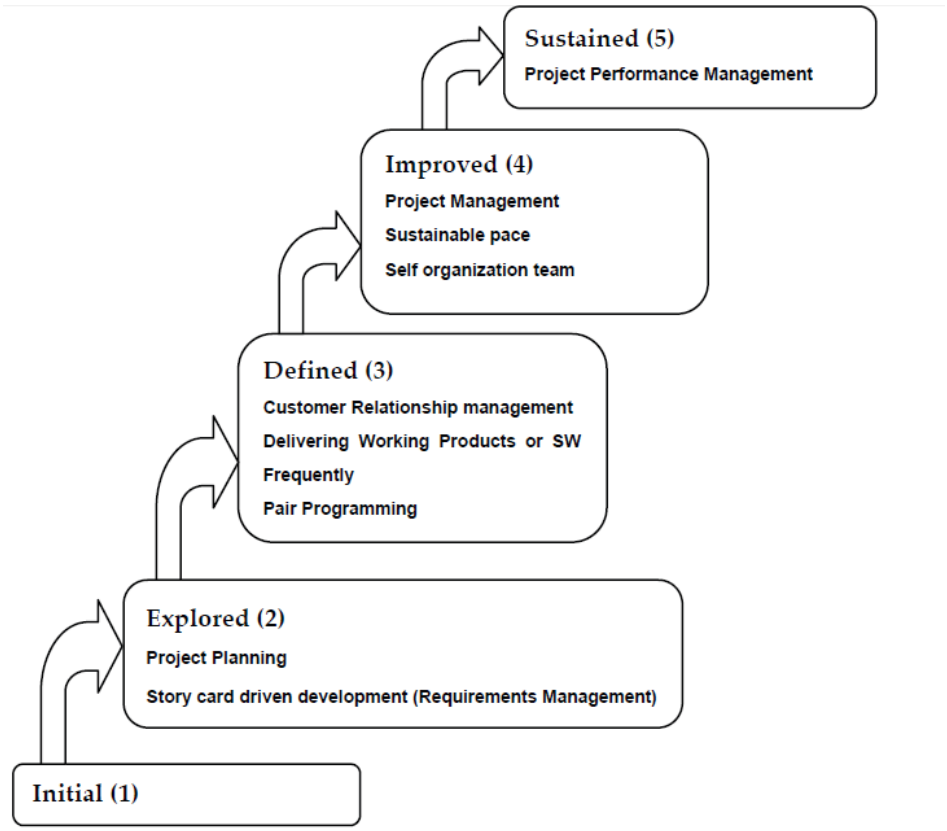

Figure 2: 5 levels of AMM

As depicted above, this model consists of five levels. The first level is the initial, at this point there is no clearly defined agile software development process and most of the practices are very slim and non-repeatable. According to this model, the main problems at this level relate to overtimes, schedule slips, communication, software quality and development costs. Success at this level depends on certain people that play a very important role in the team or organization.
Level two (explored) has a more structured software development practices than level one. The main problems at this level are communication, coding standards, overtime and customer satisfaction. According to the model, in order to complete this level and move to the next level, the following goals should be achieved:

- Project planning
- Improve agile requirements engineering
- Customer and stakeholders' orientation practices
- Enhance value, collaboration and planning practices

At this level, the current processes can be assessed and the development team will analyze these processes to identify the weaknesses and improvement areas. Learning from previous failures and successes can help the team to improve and address these issues.

Level three (defined) establishes a platform where customer relationship is very important and well maintained. The crucial aspects of this level are customer relationship, coding standards, frequent deliveries, testing, software quality, pair programming and communication. If these aspects are mastered, the organization can find itself at defined level. The problem that exists at this level are overtime, no controllable development pace for the teams and project management. At this level, there are hardly any technical issues, however, not all the problems related to the teams remain unsolved.
The goals to be achieved for this level are:

- Customer satisfaction
- Communication improvement

- Software quality
- Enhancement of coding practices and coding standards

At level four (improved), the focus on is on people and project management practices. At this level, organizations are able to collect data related to their practices and product quality. Self-organizing and empowered teams are part of this level. The teams come up with initiatives and take responsibility instead of giving to them. The teams conduct a proper risk assessment and focus on smart work instead of hard work.
The goals to be achieved for this level are:

- Empowered team and rewards
- Project management
- Risk assessment
- No overtime
- Simplicity

At level five (mature level), the focus area is on continuous improvement. There is a wealth of data related to the processes, product quality that can be analyzed in order to improve the current processes. These data can be used for defect prevention. Testing new ideas and technology are also part of this level. At this point not only the customer satisfaction is addressed, but also the developer's satisfaction. Project performance is being measured and improved through the collected data. The goals to be achieved at this level are:

- Context improvement
- Uncertainty management
- Tuning project performance
- Defect prevention

2.3.4 The agile maturity map (AMM)

The agile maturity map is a model that assists agile teams to change their mindset regarding agile development and achieve goals in a better and structured manner (Packlick, 2007). This model has been created closely with multiple teams in order to gain a better understanding what the perceptions of teams are regarding agile processes and agile practices. According to the model, it will help the teams to overcome the plateau in realizing the full potential of agile development. AMM will aid teams to improve and gain a higher maturity step by step. Furthermore, the findings from the paper suggest that: "team members value and respect a process that works and do so far more rather than having something imposed to them" (Kunz, Dumke, & Schmietendorf, 2008).

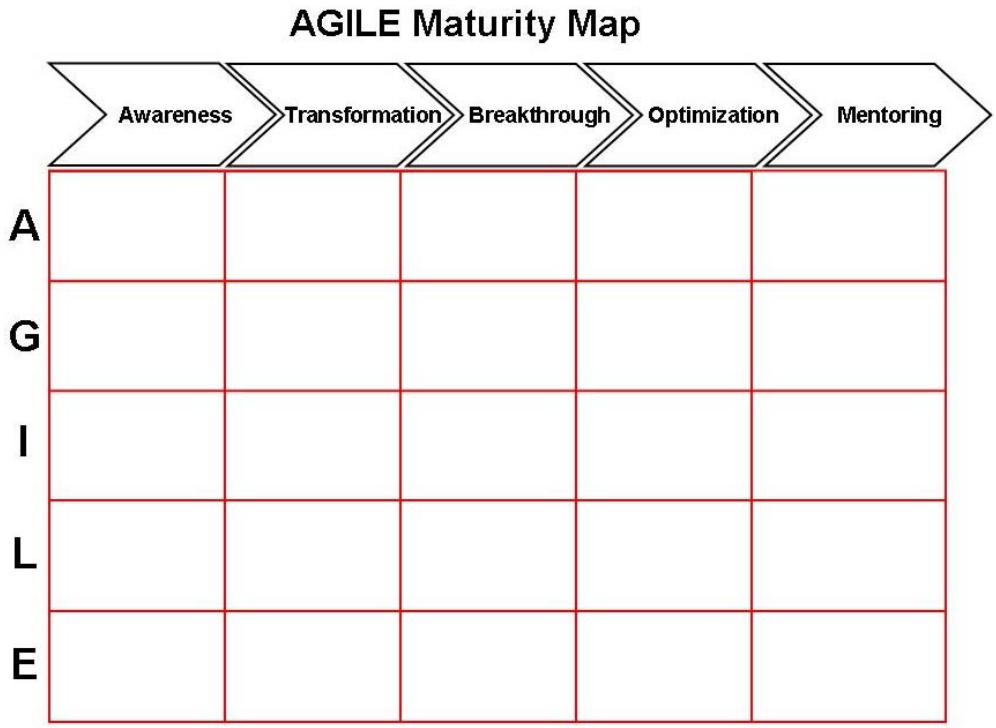

Figure 3: Agile maturity map (AMM)

AMM consist of five levels, each level represents the current state of an agile team. This model is highly goal oriented and does not dive into detailed agile practices. Each level of the model can be filled with user stories and each level represents one of the different stages of learning a team progresses through each of the five AGILE goal areas. This model is in fact a roadmap that displays the goals to be achieved by the teams and the progress of it.

Level one is the awareness. At this level, the team has an understanding of the goals to be achieved and their value. Awareness of "better" existing practices around the goals exists. Basic activities are conducted to address the goals with their related acceptance criteria. Level two is related to transformation of knowledge into practice. The theoretical knowledge is used to gain a better understanding of practices and making use of them. A clear commitment both from the leaders and the team is present at this point. The task are estimated and are broken down into smaller pieces which are easier to implement.

After the transformation, at level three the teams are working towards a breakthrough. The goals with the related acceptance criteria are achieved through consistently using agile practices. The teams are ready to break the barriers and work towards the adoption of agile practices. At this point there is significant improvement in productivity. Communication with the end-user (customer) has been increased. Developer's satisfaction has increased and automated builds have been implemented. The retrospectives are now taking place more regularly with effective implementation of outcomes. At level four optimizing, continuous improvement is taking place in order to achieve the goals. There is a clear indication that team members are creating innovative improvements. This level has no end state and is continuously implemented to ensure improvements.
The last level is related to mentoring. High performance teams have the responsibility to mentor and coach teams at lower levels in order to help them achieve higher levels. This process is conducted organization wide and is to ensure a higher level of improving software engineering.

2.3.5 A maturity model for software development organizations

The paper written by Furtado Soares and Lemos Meira provides a guide to set up and run agile methodologies based on Capability Maturity Model Integration (CMMI) (Soares & Meira, 2013). This model consists of five levels, each level provides the current state of an organization. This model is very abstract and compared to other maturity models it does not provide any detailed description of agile practices. However, every level does contain a brief description in order to understand the current state of an organization at a high level. This model will help software organizations achieve a higher rate of success when agile development values, principles and practices are adopted (Soares & Meira, 2013). According to the paper, this model contains five levels that are described below:

Level 1: initial stage where organizations do not use any methodology and their processes are unpredictable and reactive;

Level 2: the stage where processes are characterized by project. There are processes for planning and monitoring a project, but the organization's vision is by project, i.e., there is no portfolio management of projects. At this level of maturity, setting up agile methodologies starts with Scrum (a focus on managing projects and prioritizing requirements) and a part of the methodology of FDD;

Level 3: the stage where the processes are well defined and characterized by the Organization. There is a standard process with well-defined criteria to instantiate them at every context of a new project. Engineering processes are implemented with the focus on XP, FDD and Kanban;

Level 4: the stage where the processes are managed quantitatively with the focus on the agile metrics defined in Kanban and FDD;

Level 5: the stage where the process is often optimized, with the focus on continuous improvement of the processes using the principles of Lean Software Development.

2.3.6 Agile adoption framework (AAF)

Sidky and Arthur propose a framework called agile adoption framework (AAF) that provides guidance for organizations in order to adopt agile methodologies (Sidky & Arthur, 2007). The AAF provides insight to what extent an organization can become agile and whether this agility is suited for a particular organization. This framework consists of two components: a measurement index to measure the agility and a 4-stage process that employs the measurement index that provides insight in what way agile practices can be introduced within an organization.
The measurement index used in this framework is the actual maturity model that assesses the maturity of an organization. This model consists of four components that forms the measurement index. The four components are:

- *Agile levels*
- *Agile principles*
- *Agile practices and concepts*
- *Indicators*

Each level is linked to all the principles used in the model. The levels and principles can be filled in with related characteristics. The model contains of five levels. Each level has its own characteristics and covers a different perspective. The five levels are shown below with several related characteristics:

- Level 1: Collaborative. The main aspect of this level is communication and collaboration between all stakeholders.
    - Collaborative planning

    - Empowered teams
    - Coding standards
    - Knowledge sharing
    - Working closely with customer
- Level 2: Evolutionary. Early and continuous delivery of software are the main characteristics of this level.
    - Continuous delivery
    - Tracking iteration process
    - No design upfront
- Level 3: Effective. High quality software produced in an efficient and effective way is the main aspect of this level.
    - Plan features, not tasks
    - Backlog
    - Refactoring
    - Unit test
- Level 4: Adaptive. This level covers issues related to responsiveness to change.
    - Continuous customer feedback
    - Small and frequent releases
    - User stories
    - Daily stand-ups
- Level 5: Ambient. The focus at this level is to establish a vibrant environment needed to sustain and improve agility organization wide.
    - Idea agile physical setup
    - Test driven development
    - Pair-programming

The principles used in this model are "*the essential characteristics that must be reflected in a process before it is considered agile*" (Sidky & Arthur, 2007). The model has outlined five principles that are derived from the 12 principles of the agile manifesto that characterizes agile development processes.

- *Embrace change to deliver customer value*
- *Plan and deliver software frequently*
- *Human centric*
- *Technical excellence*
- *Customer collaboration*

	Agile Principles				
	Embrace Change to Deliver Customer Value	*Plan and Deliver Software Frequently*	*Human Centric*	*Technical Excellence*	*Customer Collaboration*
Level 5 **Ambient** *Establishing a vibrant environment to sustain agility*	Low process ceremony [33, 38]	Agile project estimation [20]	Ideal agile physical setup [33]	Test driven development [11] Paired programming [48] No/minimal number of level -1 or 1b people on team [17, 15]	Frequent face-to-face interaction between developers & users (collocated) [12]
Level 4 **Adaptive** *Responding to change through multiple levels of feedback*	Client driven iterations [33] Continuous customer satisfaction feedback [35, 42]	Smaller and more frequent releases (*4-8* weeks) [35] Adaptive planning [33] [20]		Daily progress tracking meetings [6] Agile documentation [39, 31] User stories [21]	Customer immediately accessible [15] Customer contract revolves around commitment of collaboration [26, 35]
Level 3: **Effective** *Developing high quality, working software in an efficient an effective manner*		Risk driven iterations [33] Plan features not tasks. [20] Maintain a list of all features and their status (backlog) [31]	Self organizing teams [33, 38, 31, 18] Frequent face-to-face communication [38, 18, 13]	Continuous integration [33] Continuous improvement (refactoring) [31, 12, 24, 5]. Unit tests [28] 30% of level 2 and level 3 people [17, 15]	
Level 2: **Evolutionary** *Delivering software early and continuously*	Evolutionary requirements [33]	Continuous delivery [33, 31, 26, 12] Planning at different levels [20]		Software configuration management [31] Tracking iteration progress [33] No big design up front (BDUF) [4, 12]	Customer contract reflective of evolutionary development [26, 35]
Level 1: **Collaborative** *Enhancing communication and collaboration*	Reflect and tune process [35, 42]	Collaborative planning [38, 18, 33]	Collaborative teams [45] Empowered and motivated teams [13]	Coding standards [29, 47, 36] Knowledge sharing tools [33] Task volunteering [33]	Customer commitment to work with developing team [13]

Figure 4: Agile Adoption Framework

2.3.7 Scrum maturity model (SMM)

Scrum maturity model focusses completely on the scrum approach. According to (Schwaber & Sutherland, 2013), scrum is a framework which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Scrum maturity model (SMM) is developed to assist organizations with process improvement, encourage self-improvement and adopting scrum on a stage and incremental approach (Sidky & Arthur, 2007). This model introduces five levels of maturity for scrum, each with its perspective goals, objectives and suggested practices. SMM is constantly aligned and renewed with best practices such as CMMI.

The first level of SMM is *initial.* At this level, organizations don't have any specific goals for process improvement, and there is no explicit definition of agile development with scrum. The problems that exists at this level are:

- Overtime
- Over-budget projects
- Poor communication

- Unsatisfactory quality

Organizations that find themselves at this level are highly dependent on skilled individuals instead of skilled teams. They operate in their own unique way and lack of having capable and skilled teams.

Level two of SMM is called managed and organizations. At this level, team has a better understanding of scrum processes. Organizations at this level, practice scrum meetings such as daily scrum and sprint planning. Furthermore, there is a clear definition of scrum definitions and roles. However, there is no indication that these practices are executed correct and effectively, therefore there might be need for process improvement. Backlog management is a part of this level and aspects such as communication with the customer, meeting deadlines, budget and schedule remain areas of improvement.

Level three of SMM is labeled as defined. This level has two focus points; customer relation management and iteration management. For the customer relation management there are three objectives to be achieved.

- Definition of “done” exists
- Product owner available success
- Sprint review meetings

The main objective of customer relation management is to maximize the collaboration with the customer. As for the iteration management, the goal is to establish a satisfactory level for the customer by delivering projects on time and within the budget. Level three will increase the success rate of the projects, however, there will be lack of standardized management.

Level four is called quantitatively managed and the focus is on standardized project management and process performance management. At this level, organizations strive to standardize all the development process for all the projects and deliver high quality products and performance levels. The second goal of this level is process performance management, the emphasis is on monitoring of all suggested practices up to level 4 of scrum maturity. Monitoring practices will give insight on the actual performance and will highlight the areas of improvement. Most of the projects are successful at this level, there is only need for improvement of the current processes.

The last level of scrum maturity is called optimizing. Organizations that are at this level have top performing teams that focus on continuous self-improvement and achieving high customer satisfaction. The main goal related to this level is performance management. This goal has four objectives to achieve:

- Successful daily scrum
- Successful sprint retrospective
- Casual analysis and resolution
- Positive indicators

2.4 Quality

2.4.1 Software quality

In this section we define quality and software quality metrics for clarity. In section 2.5, we will review the papers based on these definitions.

Quality in software development is focused on satisfying the customer's need for the software product (Sfetsos & Stamelos, 2010).
According to (ISO 8402, 1986), quality is defined as: '*The totality of characteristics of a product or service that bear on its ability to satisfy stated and implied needs'*.
The Institute of Electrical and Electronics Engineers (IEEE), defines quality as '*the degree to which a system, component, or process meets specified requirements and customer/user needs or expectations*' (IEEE, 1998).

The ISO/IEC 9126 is considered as the most widespread standards and it embraces both quality models and metrics (Botella, et al., 2004). The ISO/IEC 9126 makes a distinction between the external and internal quality and is constructed with a set of characteristics each with corresponding sub characteristics and associated metrics. The standard provides a framework for organizations to specify the target values for specific quality metrics (Sfetsos & Stamelos, 2010).

According to (Imreh & Raisinghani, 2011), quality and emphasis on quality are must have ingredients for an organization to be successful. Quality plays an imperative role in IT environments. In order to achieve high quality, organizations need to find out what quality methods and software development methods can contribute to a higher quality. Currently, agile methodologies seems to be able to provide an answer to that. The research conducted in this area suggests that agile methodologies have a positive impact on quality comparing to other software development methods (Imreh & Raisinghani, 2011; (Kumar & Bhatia, 2012; Ambler, 2005; Jyothi, Srikanth, & Rao, 2012; Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008). Therefore, with use of agile development methods organizations can increase quality of their products and processes. In this study, we focus on quality metrics used in software development.

2.4.2 Quality metrics

The ISO/IEC 9126 consist of four parts: quality model, external metrics, internal metrics and quality in use metrics (Sfetsos & Stamelos, 2010). The external metrics are associated with running software and the internal metrics are statics measures and non-related to software execution. And quality in use metrics is related to aspects when the system is running in a live environment.

Software quality metrics are categorized into: product metrics, process metrics and project metrics (Kan S. H., 2002). Product quality metrics are related to mean to time failure, defect density, customer problems and satisfaction. Process metrics are implemented to improve development and maintenance, examples are effectiveness of defect removal during development, response time of the fix process (Kan S. H., 2002). Project metrics are related to project characteristics and execution, these include cost, schedule, number of staffing and productivity.

2.5 Literature - Software Quality metrics

In this section we describe the reviewed literature with regards to software quality. Each paragraph contains a title which describes the title of the paper and the author. First, we provide an overview of all papers and the described metrics converted in five categories. Then, in the following sections, every paper will be reviewed in detail. As a result, an overview is developed with all the metrics discussed in literature, this overview can be found in the Appendix, section D.

2.5.1 Overview of metrics

The metrics have been divided in five categories as described in table below, the corresponding author of the paper where the metrics are described is shown on the vertical axis.

The category product quality relates to defects in general. Examples are; defects found during production or testing, defect arrival patterns.

Code quality metrics are related to specific code measures, such as code complexity, number of classes, lines of code.

Customer related metrics are aspects concerned with customer satisfaction, complains in terms of defect reporting and metrics such as fix response time.

Testing metrics measure aspects such as number of test cases, test success rate, number of acceptance tests.

Finally, before and after release metrics category, are metrics such as defects found before the release, defects reported by the customer and defects coming from previous release.

Paper/book	Product Quality (defects)	Code quality	Customer related metrics	Testing metrics	Before and after release metrics
Moser et. al	X				
Sfetsos & Stamelos	X	X	X	X	X
Cheng & Jansen	X			X	X
Quality in agile world		X			
Yael Dubinsky et. al	X				
Walter Ambu et. al		X			
Danilo Sato et. al		X			
H. Kan	X	X	X		X

Table 4: Overview of categorized metrics

2.5.2 Review of literature

In this section we describe the reviewed literature in detail. Every review starts with the title and the author of the paper or book.

A case study on the impact of refactoring on quality and productivity in an agile team by Moser and Abrahamsson

The case study performed by (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008) is related to the impact of refactoring in software development. Refactoring is a part of agile development that stands for continuous improvement and improving quality. According to (Fowler, 2000), refactoring

is: “a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior”. This paper mostly focuses on the assessment of the effect of refactoring on some quality characteristics that are related to software maintainability. According to (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008), only few empirical studies analyze the impact of refactoring on code quality. According to (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008), refactoring provides the following advantages:

- Refactoring helps developers to program faster
- Refactoring improves the design of the software
- Refactoring makes software easier to understand
- Refactoring helps developers to find defects

Mainly the last three advantages are imperative to the topic of software quality metrics. The impact of refactoring in this study is concerned with internal product metrics. The metrics are used to measure the typical quality attributes such as complexity, coupling and cohesion. External quality metrics such as number of defects are required to better understand and generalize the findings of this study. The study suggests that there is need of hard data from the industry verify a better and deeper understanding of the effect and impact of refactoring.

Furthermore, the metrics used in the study are selected based on generally accepted both by practitioners and researchers, and in addition used in several previous studies (Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008). The process and product metrics that are part of the research conducted in this study are shown in the table below.

Metric	Level	Definition
CBO	Class	Coupling Between Objects
LCOM	Class	Lack of Cohesion in Methods
WMC	Class	Weighted Methods per Class
RFC	Class	Response For a Class
LOC	Method	Number of Java source code statements per method
Effort	Method	Time in seconds spent for coding a method

Table 5: Software quality metrics

Empirical Studies on Quality in Agile Practices: A Systematic Literature Review by Sfetsos & Stamelos

In agile practices, quality is built into the products through a combination of best practices that provide a different perspective on quality management (Sfetsos & Stamelos, 2010). This study is a systematic literature review and its main purpose is to provide an evaluation according to ISO/IEC 12207 and ISO/IEC 1926 standards and present the empirical findings. The study has selected 123 articles that seemed to be relevant according to the research method and 46 of them were empirical studies that focused on the agile methods such as test driven development (TDD), pair programming and other agile methods. This concludes that the results of this study are based on the quality of the aforementioned agile methods.

The results related to external quality in TDD that were conducted as an experiment showed that, external quality was usually measured by:

- Number of acceptance tests
- Total number of defects
- Number of defects/KLOC

In another type of studies such as case studies or mixed studies (experiment and case study) the external quality was usually measured by:

- Number of the defects found before the release
- Number of defects reported by the customer

The findings related to external quality showed that case studies provided a strong improvement in external quality than in experiments.

The findings related to improvement in internal quality are not consistent and vary. In some cases improvements have been acknowledged related to decrease in code and design complexity in smaller units and code reusability has been increased. While in other cases, no significant differences has been experienced. Internal quality was usually measured by using these code metrics:

- Code size
- Cyclomatic complexity
- Coupling and cohesion

Pair programming has led to significant improvements. The quality has been increased in the following areas:

- Design and code quality
- Teamwork
- Communication
- Code spreading and understanding
- Information and knowledge transfer
- Efficient programmers

Other agile methods included XP (extreme programming) and scrum. Applying agile practices such as planning game has led to a better work estimation and quality has been increased using refactoring. Improvement in customer satisfaction has been reported as well.

Controlling and Monitoring Agile Software Development in Three Dutch Product Software Companies by Cheng & Jansen

This paper is based on a case study of three Dutch software companies and it describes what the necessary measurements and actions are in order to steer the development process successfully. According to (Cheng, Jansen, & Remmers, 2009), successful steering is reached by using Key Performance Indicators (KPI) and interventions.

A list of KPI's and interventions are constructed to measure and monitor the development process. The KPI's will provide measurements based on 4 categories; teams, person, task and quality. The KPI's that will provide insight related to quality are the following:

- Total reported defects
- Number of critical defects
- Outstanding defects
- Fixed/solved defects

- Defects coming from previous release
- Hours spent on bug
- Test success rate
- Test failure rate

The metrics mentioned above are both based on internal and external quality and are used to provide the managers and teams with useful information regarding the quality of their products. Interventions are used when the KPI's indicate that certain goals are in danger. At this point managers can intervene by using the interventions that are related to the KPI's. Interventions that are related to quality KPI's are in place to create the awareness of the quality of the software. These interventions are:

- Set criteria for working software
- Let customers test the software
- Make a visible chart for the whole organization

Furthermore the paper suggests that more work is required for the extension and validation of the list of KPI's and interventions.

Quality in agile world

This paper is an introduction to common agile software development methods and implies that agile development leads to software of a much higher quality than the traditional method (Ambler, 2005). According to (Ambler, 2005), "It is common to say that agilist are quality infected" and the role of the quality professionals has changed.

Many of the agile development techniques are focused on delivering and creating high quality software. According to this paper, these techniques and concepts that contribute to higher quality are:

- Refactoring
- Test-driven development (TDD)
- Tests replace traditional artifacts
- Agile model driven development (AMDD)

Refactoring is a method that is used in many agile development techniques. By implementing refactoring, small changes are applied to the source code to improve its design and make it easier to work with (Fowler, 1999).

The paper suggests that there are several implications for quality professionals related to agile software development. These implications include:

- Greater quality implies less need for quality assurance activities
- Get used to "incomplete" artifacts such as models, documents, source code
- Become a generalizing specialist

Agile Metrics at the Israeli Air Force by Yael Dubinsky, David Talby, Orit Hazzan, and Arie Keren

Accurate metrics are the essential aspects in order to take professional decisions (Dubinsky, Talby, Hazzan, & Keren, 2005). The study performed by (Dubinsky, Talby, Hazzan, & Keren, 2005) focuses on

implementation of extreme programming into a software development team of Israeli's air force. The study takes into account the metric mechanism that was established during the kick-off of the project and uses a subset of metrics to measure the implementation process. These metrics should provide information regarding:

- Effectiveness of the process
- Decision making
- Analyze long-term effects
- Increase confidence on all management levels

The quantitative data has been gathered through the team reports and automated reports that are output by the development environment. According to the study (Dubinsky, Talby, Hazzan, & Keren, 2005), metrics can be used for three purposes:

- Communication to the team
- Faster decision making
- Reports to upper management

In general, there are four metrics that have been used to measure the implementation process of extreme programming at Israeli's air force. These metrics include:

Metric	Description
Product size	The amount of completed work
Pulse	Measures the continuous integration
Burn	Presents the remaining work versus the remaining human resources
Faults	The number of faults per iteration

Table 6: Metrics to measure implementation process of XP

Studying the Evolution of Quality Metrics in an Agile/Distributed Project by Walter Ambu, Giulio Concas, Michele Marchesi, and Sandro Pinna. 2006

This paper is based on an empirical study of agile teams that evolved into a distributed context and analyzes the development of the project. According to (Ambu, Concas, Marchesi, & Pinna, 2006), there are several studies conducted based on the experiences in applying agile practices in a distributed context, however, there are no analysis regarding the source code quality metrics. Lack of defects and maintainability are usually the measures to define the quality of a project (Ambu, Concas, Marchesi, & Pinna, 2006). The source code analysis performed in (Ambu, 2006), adopted the Chidamber and Kemerer (CK) suite that contains six metrics. These metrics are related to measuring source code quality. According to the study, the following metrics represent the CK suite:

Metric	Description
Weighted Methods per Class (WMC)	A weighted sum of all the methods defined in a class
Coupling Between Object Classes (CBO)	A count of the number of other classes with which a given class is coupled
Depth of Inheritance Tree (DIT)	The length of the longest path from a given class to the root

	class in the inheritance hierarchy
Number of Children (NOC)	A count of the number of immediate child classes inherited by a given class
Response for a Class (RFC)	A count of the methods that are potentially invoked in response to a message received by an object of a particular class
Lack of Cohesion of Methods (LCOM)	A count of the number of method-pairs with zero similarity minus the count of method pairs with non-zero similarity

Table 7: CK suite metrics

There are reports regarding the implementation of CK suite in commercial settings (Ambu, Concas, Marchesi, & Pinna, 2006). One of the reported findings concluded that applying CK suite has reduced productivity and increased the rework/design effort (Ambu, Concas, Marchesi, & Pinna, 2006). Another study reported that the measures of CK suite introduced class defect density.

Furthermore the study analyzed the JAPS project, which was initiated by several web developers. JAPS is a solution to build web portals, integrate services and handling content through content management system (CMS). In the development process of JAPS, the research group selected the CK suite metric and an extra set of metrics to analyze. The extra set of metrics is shown in the table below.

Metric	Description
Number of Classes	Total number of classes
Class Size	The size of a class has been measured by counting the lines of code (LOC)
Number of Test Cases	The number of test cases may be considered as an indicator of testing activity
Number of Assertions	The number of test methods

Table 8: JAPS metrics

The CK suite quality metrics were used to monitor the project. By applying the metrics, final findings conclude that initially the system complexity has been increased. However, after applying effective implementation of agile practices, the systems were simplified. Furthermore, the teams were unable to improve all the metrics to the same extent.

Tracking the Evolution of Object-Oriented Quality Metrics on Agile Projects by Danilo Sato, Alfredo Goldman, and Fabio Kon. 2007

The study conducted by (Sato, Goldman, & Kon, 2007) is based on an analysis of seven projects that tracks and evaluates the evolution of Object-Oriented (OO) quality metrics. From these seven projects, there are five projects that have been executed in an academic environment and the other two are governmental projects. Most of the projects were implementing agile methods (XP) and some of them introduced it later.

This paper has reviewed other studies that claim the use of object-oriented quality metrics can aid developers to understand complex design, detecting design flaws and preventing defects. The metrics that have been used to analyze the source code are a mixture of CK's suite (Ambu, Concas, Marchesi, & Pinna, 2006) and from Martin's suite (Sato, Goldman, & Kon, 2007). In addition to aforementioned metrics, the study also chose for extra metrics (LOC and v (G)).

Metrics analyzed in this study are listed below. The CK's metrics have been already described in the former study (Ambu, Concas, Marchesi, & Pinna, 2006) and table 7**.**

- Weighted Methods per Class (WMC)
- Depth of Inheritance Tree (DIT)
- Lack of Cohesion of Methods (LCOM)
- Number of Children (NOC)
- Afferent Coupling (AC): the total number of classes outside a package that depend on classes inside the package.
- Efferent Coupling (EC): the total number of classes inside a package that depend on classes outside the package
- LOC: the total number of non-blank, non-comment lines of source code in a class of the system
- McCabe's Cyclomatic Complexity (v(G)): measures the amount of decision logic in a single software module

According to (Sato, Goldman, & Kon, 2007), several studies have shown that classes with higher LOC and WMC are more error prone. In one of the projects that was analyzed by (Sato, Goldman, & Kon, 2007), there was a decrease in size and complexity of the source code. This study concludes that the decrease in size and complexity is related to automated tests and refactoring. In addition, based on the other six projects that were analyzed in this study, the conclusion is that projects with less agile practices have resulted in higher size and complexity, more error prone required more testing and maintenance effort.

Metrics and Models in Software Quality Engineering by Stephen H. Kan, 4th chapter: Software quality metrics overview

The fourth chapter of the book (Kan, 2002) provides an overview of software quality metrics that are used in software development. The focus of software quality metrics can be divided into three subjects: quality of the product, process and project. According to (Kan, 2002), the aforementioned metrics can be grouped into three categories: end-product quality metrics, in-process quality metrics, and maintenance quality metrics. Table 9 summarizes the categorized metrics and the related description.

Metric	Description
Product quality	
Mean time to failure	Measures the time between failures
Defect density	Measures the defects relative to the software size (lines of code, function points, etc.)
Customer-reported problems	The problems customers encounter when using the product
Customer satisfaction	Measures customer satisfaction through survey data based on a five-point scale: Very satisfied, Satisfied, Neutral, Dissatisfied, Very dissatisfied
In-process quality	
Phase-based defect removal pattern	The pattern of phase-based defect removal reflects the overall defect removal ability of the development process
Defect removal effectiveness	Measures the effectiveness of defects both during the development and defects latent in the product
Defect density during formal machine testing	Defect rate during formal machine testing
Defect arrival pattern during formal machine testing	Measures the pattern that related to defect arrivals that stabilize at a very low level, or times between failures that are far apart, before ending the testing effort and releasing the software to the field

Maintenance quality	
Fix backlog	Fix backlog is a workload statement for software maintenance, it's a simple count of reported problems that remain at the end of each month or week
Backlog management index	Provides a ratio of the number of closed, or solved, problems to a number of problem arrivals during the month
Fix response time and fix responsiveness	It measures the agree-to fix time and the ability to meet one's commitment to the customer
Percent delinquent fixes	For each fix, if the turnaround time greatly exceeds the required response time, then it is classified as delinquent
Defective fixes	A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect

Table 9: Metrics based on three categories

In addition to discussed metrics, according to (Kan, 2002), the list below is only related to quality metrics.

- Overall customer satisfaction as well as satisfaction with various quality attributes such as CUPRIMDS (capability, usability, performance, reliability, install, maintenance, documentation/information, and service)
- Post-release defect rates
- Customer problem calls per month
- Fix response time
- Number of defective fixes
- Backlog management index
- Post-release arrival patterns of defects and problems (both defects and non-defect-oriented problems)
- Defect removal model for the software development process
- Phase effectiveness (for each phase of inspection and testing)
- Inspection coverage and effort
- Compile failures and build/integration defects
- Weekly defect arrivals and backlog during testing
- Defect severity
- Defect cause and problem component analysis
- Reliability: mean time to initial program loading (IPL) during testing
- The stress level of the system during testing as measured in level of CPU use in terms of the number of CPU hours per system per day during stress testing
- Number of system crashes and hangs during stress testing and system testing
- Models for post-release defect estimation
- Various customer feedback metrics at the end of the development cycle before the product is shipped
- S curves for project progress comparing actual to plan for each phase of development such as number of inspections conducted by week, LOC integrated by week, number of test cases attempted and succeeded by week, and so forth.

3 METHODOLOGY

3.1 Research design

The research was primarily based on data collection of multiple agile organizations. The first step is to review the relevant literature to construct an understanding of the topic. Next, the literature will be reviewed to prepare the survey and interview questions. Then we will perform a multiple case study in agile organizations using the survey and interview questions. Due to the number of desired organizations and people to research, a fast and simple data collection method is required. Survey is the best approach because it requires less time to conduct and offers a variety of choices and simplicity. The rest of the questions which are mainly the open questions, are constructed in an interview form. The goal of the interviews and surveys is to provide a more in depth knowledge and information related to certain agile activities and practices. The figure below depicts the approach.

Figure 5: Research design

3.2 Literature review

The literature has been conducted with the focus mainly on agile maturity models and software quality metrics. We have used Google scholar and University Library Catalogue to find the relevant literature. These sources provided access to papers, articles and books related to the topic of this study.

We have used the following keywords:" agile maturity", "agile maturity models", "software quality", "agile quality metrics", "quality metrics". Based on the outcome we have selected the most popular papers based on the relevancy to the topic. We aimed to select only papers, articles and books that have been published since 2005 to avoid very outdated information related to the topics of agile maturity and software quality metrics.

3.3 Research question design

The figure below shows how we constructed the interview and survey questions for the case study. In order to construct the questions both for the survey and the interviews, we need to analyze the sub-research questions. The first step is to identify what the necessary questions are that can provide an answer to a sub-question. Following this approach, a set of questions is constructed that are related to a research sub-question. The total set of questions will eventually aid to answer the sub-question. As a result, based on the research questions, we will use the relevant literature to construct Interview and survey questions are constructed with the aid of the relevant literature.

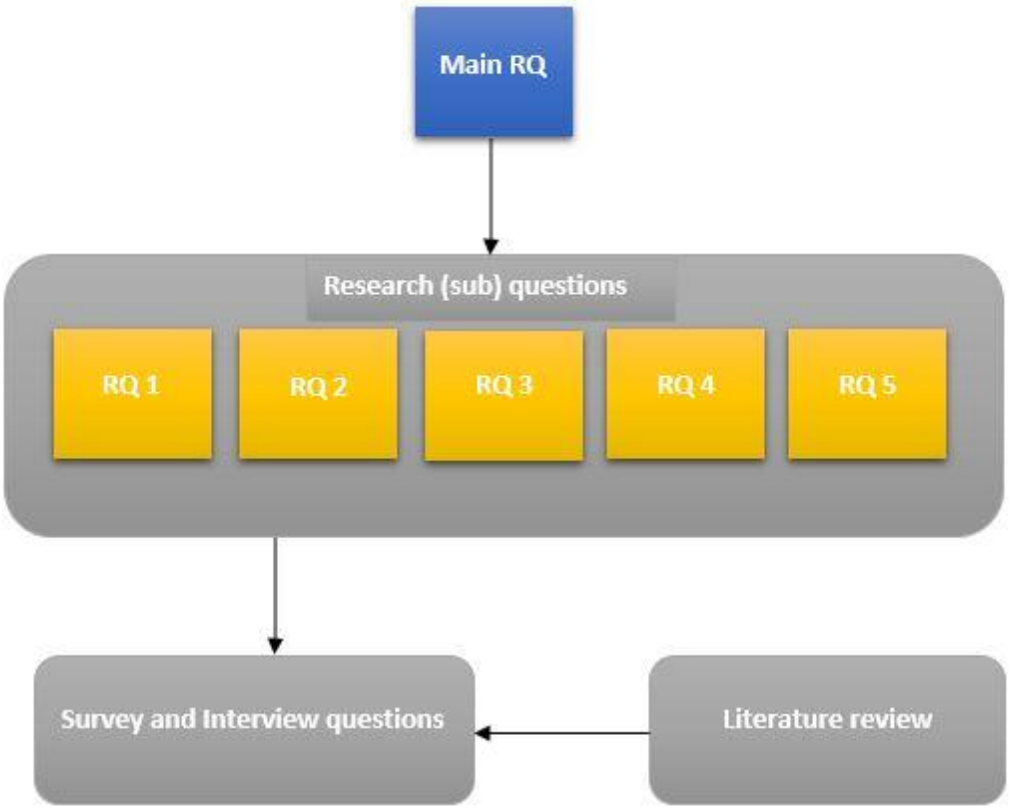

Figure 6: Research questions design

In the table below we present the survey and interview questions related to specific research question. The research questions are displayed in bold and indicated as "RQ". In addition, the relevancy of the questions is described, and what possible variable can come out of it. This variable is used later for data analysis.

Table definition:

- #: question numbers
- Interview/survey: indicates whether the question is used in the survey or the interview
- Question: lists all the questions implemented in survey and interviews
- Why: elaborates why this question is relevant to the research
- Variable: indicates the form of the expected output that can help to answer the research question

Abbreviations:

- QM: Quality metric
- RQ: Research question
- Dev: Developer
- QA: Quality assurance

Q#	Interview/survey	Question	What does it contribute?	Variable
A	**RQ**	**What are the most effective quality metrics (QM) that are being used in agile organizations?**		
1	Survey – Open question	What measures or metrics do you collect? Please also specify all the measures that you take but are not on this list	More QM increases chance finding effective QM's	#metrics and popularity
2	Survey – multiple choice	How do you measure software quality?	Verifies if QM's are correctly collected and are valid	Using tools, Manually, None
3	Interview	What are the most important quality metrics? Why these?	Important QM's can be effective	List of metrics
4	Interview	Do you change the quality metrics often? Why?	Changing QM's can help to find better QM's	Yes/No
5	Interview	When is a quality metric effective?	Effective QM's will be used more than other QM's	Customer satisfaction
6	Survey – Open question	How much is the source code covered by unit testing?	Higher coverage leads to less defects	Coverage
		Verification questions		
7	Survey – Scale 1-7	Does the code often need maintenance?	Use of effective QM can increase code quality	
8	Survey – Scale 1-7	Is there any "extra time" given for cleaning up and re-factoring the source code?		
9	Survey – Scale 1-7	Is the test engineer always testing the latest build?		
B	**RQ**	**What are the most effective agile practices that support maturity?**		
10	Survey – Multiple choice	What agile practices does your team apply? (List)	Popular practices are used the most, therefore can be effective	#practices
11	Interview	Is there a continuous delivery pipeline? How does it look like?	Continuous delivery belongs to higher level of agile maturity	Yes/No, to what extent(initial-mature)
12	Survey – Scale 1-7	Do the test engineers make use of automated test scripts?	Automated tests can define higher agile maturity	Yes/No
13	Survey – Open question	What percentage of test scripts are automated?**(Link to A)**	More automated tests can increase quality	Percentage
14	Interview	What agile practices are really necessary?	Highlights Important practices	List of practices
		Verification questions		
15	Survey – Open question	Please specify all the participants in the planning session (e.g. Dev. QA, info analyst etc.).	Validates if all relevant roles are attending	#roles
16	Survey – Scale 1-7	All members of the team actively participated during iteration planning meetings	Validates if all team members are attending the planning	All team members or some
17	Survey – Scale 1-7	All the tasks for the sprint were estimated	If the team works with estimation	

18	Survey – Scale 1-7	The team never missed the sprint deadline		
19	Survey – Scale 1-7	Working software was the primary measure of project progress	Link to A	
20	Survey – Scale 1-7	The team rather reduced the scope than delayed the deadline		
21	Survey – Scale 1-7	At the end of the iteration, we delivered a potentially shippable product		
22	Survey – Open question	How frequently do you release working software? E.g. Weekly, monthly.		
23	Survey – Scale 1-7	Scrum master was always present during the stand-up.		
24	Survey – Scale 1-7	Stand up meetings were extremely short (max. 15 minutes)		
25	Survey – Scale 1-7	All relevant technical issues or organizational impediments came up in the stand-up meetings		
26	Survey – Scale 1-7	In the retrospectives (or shortly afterwards), we systematically assigned all important points for improvement to responsible individuals		
27	Survey – Scale 1-7	The team was always sitting together in the same room		
C	**RQ**	**How do organizations measure maturity of their agile software development?**		
28	Survey – Open question	What is currently the focus point of the organization to improve on agile?	Shows awareness of the current state and next steps to improve (high level)	List of goals(strategic)
29	Interview	What is the area of improvements for you and your team to use agile methods better?	Shows awareness of current state and next step to improve(lower level)	List of improvements
30	Interview	Is the company performing any agile assessment? How? What measures?	Provides metrics to measure agile maturity	#assessments, #measures, List of measures
31	Interview	How familiar are the team members with agile methods? Experience in years?	More experience in agile could lead to higher maturity	#Years
32	Survey – Open question	Do any of the team members have any form of agile certification? E.g. SM, Exin Scrum etc.	Certification can be metric	#certifications
33	Survey – Open question	What agile practices/techniques would you like to conduct that are currently not in place?	There is understanding of current state and limitations	#practices
34	Interview	Is there any agile training/workshop provided by the company? How often? For who?	There is understanding of the current state and limitations, improvement is needed	#trainings/workshops
		Verification questions		
35	Survey – Scale 1-7	Is there any freedom by the organization to allow implementation of agile practices?		

36	Survey – Open question	How much does the team make use of this "freedom" to implement (new) agile practices?		
D	**RQ**	**What is the impact of successful or failed projects on quality metrics? What factors do influence this?**		
37	Survey – Open question	How long after the sprint ends, you receive feedback from the customer?	Info about project success or failure	Time
38	Interview	What feedback do you receive regarding the quality?	Verifies if good quality was delivered, else quality metrics were not good	Customer satisfaction
39	Survey – Open question	What percentage of projects is successful?	High percentage has less effect on quality metrics	#%
40	Interview	Why is the success rate like this? **(Link to A)**	Success can be linked to use of right quality metrics	List of activities related to success
41	Interview	What measures did you take when projects failed due to poor quality?	Provides list of measures, e.g. increase #quality metrics	List of measures
42	Interview	How does the customer feedback change the test or the development process?	If poor quality was delivered, the Dev/QA process should change	Customer satisfaction
		Verification questions		
43	Survey – Open question	How often do you measure customer satisfaction?		
44	Survey – Open question	How do you measure customer satisfaction?		Quality metric
E	**RQ**	**At what maturity levels are which software quality metrics implemented?** *Can be answered better after data analysis*		
45	Interview	When do you introduce new quality metrics? Why?	Indicates at what stage new QM's are used	
46	Interview	How often do you implement new quality metrics?	New QM's can indicate continuous improvement and agile growth	Time
47	Survey – Open question	What artefacts are created specifically for people outside of the team?		

Table 10: Interview and survey questions

3.4 Case selection strategy

After finalizing the structure of the survey and interview, data collection phase can be started. For this purpose we have chosen to perform a multiple case study in organizations that are implementing agile development. We will study small, medium and large organizations. The objective is to study organizations with multiple range of experience. This means, organizations that just started implementing agile methodologies, organizations that have been implementing agile for 1-3 years and organizations that have been implementing agile for more than five years. Using this strategy, this study will look into organizations from low to high experience with agile methodologies.

3.4.1 Interview and survey

From each organization two team members will be interviewed and asked to fill in the survey. The session will start with filling in the survey and after that the interview part will start. A team member is someone that is a member of the agile team. Preferably the first choice is to interview a developer and a software tester. A developer knows the all the practices related to software development and can provide answers to technical development questions. A software tester can answer all the questions related to the testing and software quality process. To have at least two persons from each organization interviewed, will provide a more general perception of the organization, and in addition, it provides more solid data. If there is no possibility to speak to a developer or a tester, different roles within an agile team can be used.

An agile team can consist of the following roles:

- Developer
- Tester
- Product owner
- Scrum master
- Agile coach
- Information/business analyst
- Team lead
- Architect

The survey consists of 32 questions in various forms; open questions, multiple choice and answers based on a certain scale. The scale is based on the Likert scale (University of Connecticut, sd) and offers a choice between “never” and “always” with a score of 1 to 7 accordingly.

The scale is defined as follows:

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
		Not usually	***Rarely***	***Occasionally***	***Often***	***Usually***		

The complete survey and the interview questions can be found in the appendix section A and B.

3.5 Method

Due to the developments on the topic of this research, we need to base our research on data gathered from organizations in order to understand certain aspects related to agile maturity and quality metrics. The results of this study are a combination of theory, interviews and observations. Therefore a qualitative approach will be taken in this research. Qualitative research will help us understand the underlying developments and highlight important insights found in our samples.

3.6 Data analysis

In order to answer the main research question we need to categorize the maturity level of each team from every organization and perceive the related quality metrics.

It's imperative to mention that the representing team from an organization will not account for the entire organization and therefore will not represent the maturity level of the organization but solely the maturity level of a specific team within that organization. The study aims to provide a general perception of implementation of agile methodologies within an organization without trying to determine the agility of the entire organization.

We will classify the maturity scores in three categories (low, medium and high). The low maturity category consists of organizations that scored less than 150 points. Organizations in this category are considered low mature and are in the beginning stage of agile implementation. The medium maturity level is representing organizations that have implemented agile methodologies for some years, but there is no substantial growth. The score range for the medium level is 150-210. These organizations need to improve on certain areas where they scored a lower score to become high mature. The specific area to improve is presented in section 4.5.5. The high maturity level consists of organizations with a score higher than 210. As a result, we classified the studied organizations in table 11 with the corresponding maturity level.

Low maturity level	Medium maturity level	High maturity level
Score less than 150	Score between 150-210	Score between 210-301

Table 11: Maturity level calculation

3.6.1 Maturity score calculation

The adopted approach to determine agile maturity is to evaluate the scores from the survey and the interviews based on pre-determined score list. The evaluation is divided in two parts, survey and interview evaluation. As for the survey, each question can have a maximum score. In particular cases, a question can provide more points due to the relevancy and therefore will have an alternate score. The evaluation of interviews is determined differently, only a selected number of interview questions can score points. In both, the survey and the interview questions, the weight is evaluated differently. Some of the questions are more important and therefore will have a higher maximum score. As a result, the weight of all the questions is diverse and the score depends on the responses. In table 12 we have defined the score calculation.

In order to calculate a precise maturity score, we need to analyze organizations from five dimensions. These dimension are described in the table below. In addition, the table presents the corresponding questions of every dimension.

Dimension	Software quality	Testing process	Agile practices	Software deliveries	Organizational strategy
Related question	1, 2, 4, 34	3, 6, 8, 9	5, 7, 11, 12, 18, 19, 20, 21, 22, 36	13, 15, 16, 17	27, 39, 41
Total points	47	61	122	43	28

Table 12: Maturity dimensions and related questions

A complete list with questions and the corresponding maximum score to calculate is shown in table 12. This table displays only the question numbers, to find the corresponding questions, please refer to the appendix, section A and B. The questions are categorized in the five respective dimensions.

Question number	Maximum points to score	Calculation	Justification
Quality	47		
1	26	Every quality metric accounts for 2 points	More quality metrics can indicate more quality insight and increased use of agile aspects
2	7	Manually=3, automatically= 7	Automatic data collection excludes human error
4	7		Survey score 1-7
34	7		Survey score 1-7
Testing	61		
3	20	Every 5% accounts for 1 point	High unit test coverage indicates good implementation of this practice
6	7		Survey score 1-7
8	14	Scale 1-7. Score is multiplied by 2	Automated testing is important aspect of agile
9	20	Every 5% accounts for 1 point	Automated testing is important aspect of agile
Agile practices	122		
5	7		Survey score 1-7
7	52	Every practice accounts for 2 points	Use of many agile practices can indicate maturity
11	7		Survey score 1-7
12	7		Survey score 1-7
18	7		Survey score 1-7
19	7		Survey score 1-7
20	7		Survey score 1-7
21	14	Scale 1-7. Score is multiplied by 2	Good implementation of this practices indicates maturity
22	14	Scale 1-7. Score is multiplied by 2	Sitting together is very important aspect of agile
SW deliveries	43		
13	7		Survey score 1-7
15	7		Survey score 1-7
16	7		Survey score 1-7
17	15	Continuously/daily=15 points, weekly=10, monthly=5	Fast software release indicates agility

27	7		Survey score 1-7
Org strategy	28		
36	14	Started = 3, half implemented = 7, complete implementation = 14 points	Continuous delivery is an indication for agile maturity
39	7		Survey score 1-7
41	7		Survey score 1-7
Total	**301**		

Table 13: Score definition

4 RESULTS

In the following sections the results of this study are described. The data collection, which consisted of interviews and surveys, is performed in 11 organizations worldwide (Netherlands, United Kingdom, Israel), resulted in 22 interviews and 22 surveys. At the end of each section, there is a preliminary conclusion given based on the results of that section.

4.1 Participating organizations

The organizations that have pledged their cooperation and have contributed to this research are listed below. A short description is provided based on the studied environment and the general information regarding the organization.

Organization	Participants	Role	Location
A	3	Agile coach Software developer/scrum master Information analyst	Netherlands
B	3	Product owner Business analyst Software developer	Netherlands
C	2	Designer/tester Software developer	Netherlands
D	2	Scrum master Software developer	Israel
E	2	Software developer	Netherlands
F	2	Software developer Scrum master/operations	United Kingdom
G	2	Software developer Team lead/software developer	United Kingdom
H	2	Tester Architect/software developer	Netherlands
I	2	Product owner/tester Software developer	Netherlands
J	2	Scrum master/software developer Software developer	Netherlands
K	1	Software developer	Netherlands

Gibbon is a startup located in Leiden and has six employees. Gibbon has gained various investment rounds and is growing. The startup is now active for more than two years and is offering online learning content to thousands of users worldwide. Gibbon is not following a strict software development, but since its beginning, the startup has been implementing a software development method that is similar to agile.

NICE Systems (NASDAQ: NICE), is the worldwide leader of intent-based solutions that capture and analyze interactions and transactions, realize intent, and extract and leverage insights to deliver impact in real time. Driven by cross-channel and multi-sensor analytics, NICE solutions enable organizations to improve business performance, increase operational efficiency, prevent financial crime, ensure compliance, and enhance safety and security. NICE serves over 25,000 organizations in the enterprise and security sectors, representing a variety of sizes and industries in more than 150 countries, and including over 80 of the Fortune 100 companies (NICE.com, sd). NICE Israel has started implementing agile methodologies three years ago.

The Alkmaar site of NICE systems is the formerly known as Cybertech International. The leading provider of call recording solutions mainly focused on the trading floors. In 2011 NICE Systems acquired Cybertech. The top financial organizations and banks are implementing the recording solutions realized in Alkmaar. There are 80 employees working in NICE Alkmaar. Since 2010 the company has been implementing agile methodologies (Scrum).

Fizzback is a company that has been acquired by NICE Systems for $80 million in 2011 and is offering solutions for customer analytics. Fizzback sends consumers requests for feedback relating to a specific interaction or transaction via mobile, web or social media. The feedback is then analyzed by Fizzback to determine a relevant response, and the company subsequently engages the consumer at the point of experience, for example, in the contact center, branch, point of sale, mobile app, or on the Web. Fizzback is located in London and the IT development is also operating from London (TechCrunch.com, sd). Since three months ago Fizzback started implementing agile methodologies (Scrum).

Causata, Inc. is a leading provider of Customer Experience Management (CXM) software. Causata has been acquired by NICE in 2013. Built on an HBase big data architecture, the predictive analytics and real-time omni-channel offer management applications enable B2C companies to create meaningful customer experiences through data. The industry-specific applications help companies increase cross-selling and customer acquisition while reducing churn. Founded in 2009 and funded by Accel Partners, Causata's headquarters are in San Mateo, California with a development office in London, England (CrunchBase, sd). Agile methodologies (Scrum) have been introduced in Causata 1,5 years ago.

Bank Mendes Gans (BMG) is part of the ING group and is worldwide known as a niche player in the area of liquidity and information management for large organizations. BMG is one of the most important specialist in the field of cash management. BMG is located in Amsterdam and is independently operating business unit of ING commercial banking. BMG started implementing agile methodologies (Scrum) 1,5 years ago.

TomTom is a company that manufactures navigation and develops mapping products. TomTom's headquarter is in Amsterdam and has more than 4000 employees worldwide. TomTom is mostly known for its navigation products and is active in 41 countries. TomTom has been implementing agile methodologies (Scrum) since 2008.

ING is a global financial institution with headquarters based in the Netherlands. With 53,000 employees in 40 countries is ING a global player. ING is the market leader in the Benelux and has a strong position in retail and commercial banking. ING is very active in IT development and has its own software development houses where teams are working according to agile methodologies (Scrum) since 2011.

ABN AMRO is a leading bank within the Netherlands and serves clients across the globe with a comprehensive range of products and services. Internationally is ABN AMRO active in areas which the bank has substantial expertise such as private banking, energy and commodities & transportations (ECT) and clearing. Most of the IT development in ABN AMRO is outsourced to external suppliers such as IBM. One of the fewer locations of ABN AMRO that implements in house development is the software factory division. This division is responsible for the IT products that are part of the ABN AMRO's "Hypotheken Groep". The division is subject to the data collection and has been implementing agile methodologies (Scrum) for eight years already.

Bol.com is the most visited retail website in the Netherlands serving 5 million active customers. It offers a broad range of products in various non-food categories including books, entertainment, electronics and toys. The main office of Bol.com is located in Utrecht where 750 employees are working. From the IT perspective, Bol.com is generally known for its experience in implementation of agile methodologies. Bol.com has been implementing agile methodologies (Scrum) since 2008.

The Dutch tax office is a governmental organization that has about 30,000 employees, which are responsible for various activities. The IT department is responsible for the digital processing of people's assets and administrations and enabling the convenient IT environment for the citizens. The Dutch tax office has different offices across the country. The IT department is mainly based in Apeldoorn and Utrecht in the Netherlands. The Dutch tax office started implementing agile methodologies (Scrum) in 2011.

4.2 Agile maturity score

This section presents the results of the participating companies with regards to their maturity score in random order. We will not reveal the organization's name related to the results, therefore their anonymity will be preserved. In order to generate a score from all the participants of an organization, the scores will be summed up and generate an average score. The total score of the average is eventually the maturity score for the team. In the following paragraphs each organization is presented with the related maturity score. The score is outlined on the vertical axis of each figure. The horizontal axis represents the question numbers. The question numbers are based on the survey and interview question numbers used. The survey and the interview questions can be found in the appendix section A and B. The maturity is determined on the basis of 25 questions that are representing different maturity dimensions. The dimensions are listed below.

- Software quality
- Testing process
- Agile practices
- Software deliveries
- Organizational strategy on agile

Organization A

The figure below presents the results with regard to the maturity score of organization A. The data is accumulated results from two participants from the organization. Participant 1 has been working for eight years at the company and has experience with agile methodologies since she started working at this firm. Participant 2 has been working for 6,5 years at the company and has experience working according to agile methodologies since his previous job.

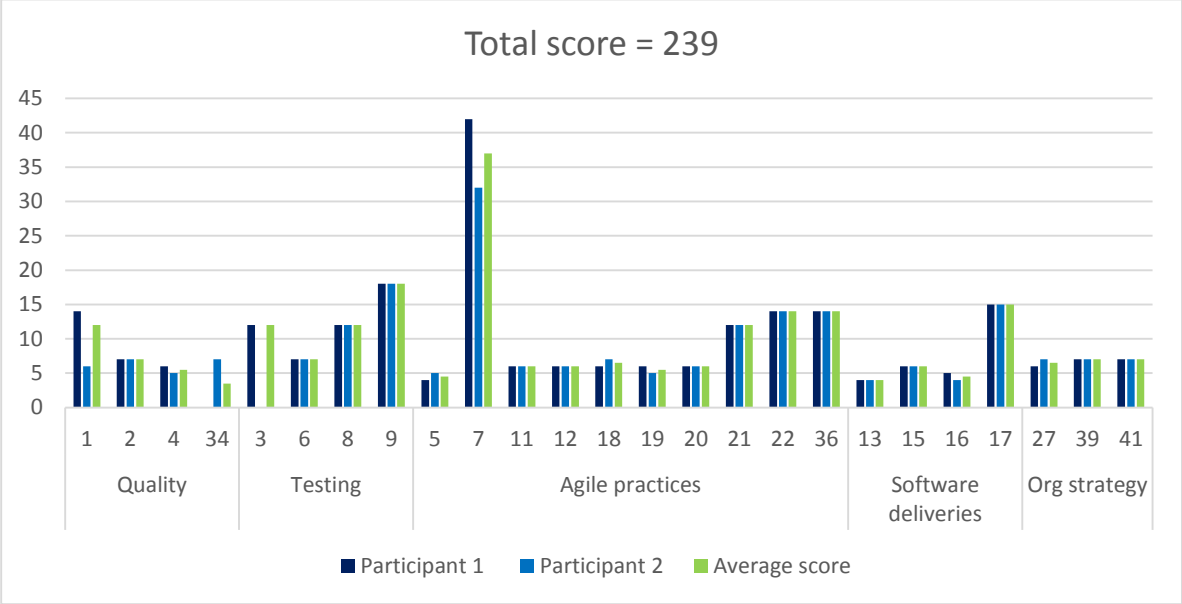

Figure 7: Organization A

Organization B

The participants in organization B have experience with agile methods for two years already. The company started implementing Scrum organization wide 2,5 years ago.

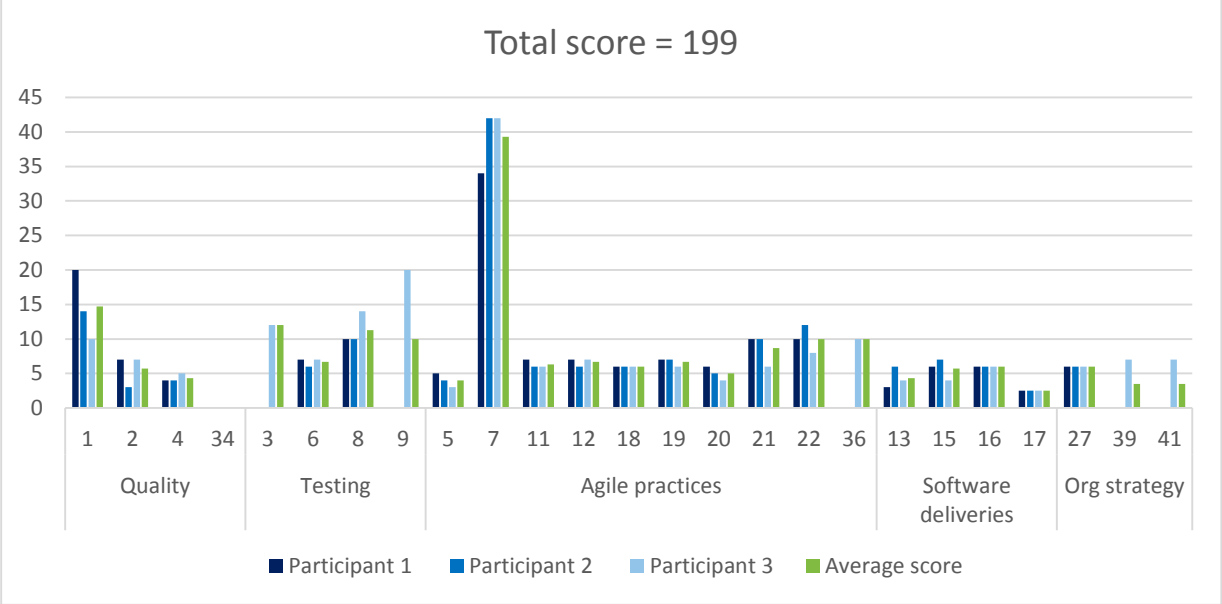

Figure 8: Organization B

Organization C

This organization is not strictly following any agile methods, but instead they make use of best practices. These best practices are combinations of Scrum, XP and a method that is generated in house and works locally. This organization is a start-up company and the employees are familiar with agile methods only from the theory without any previous experience.

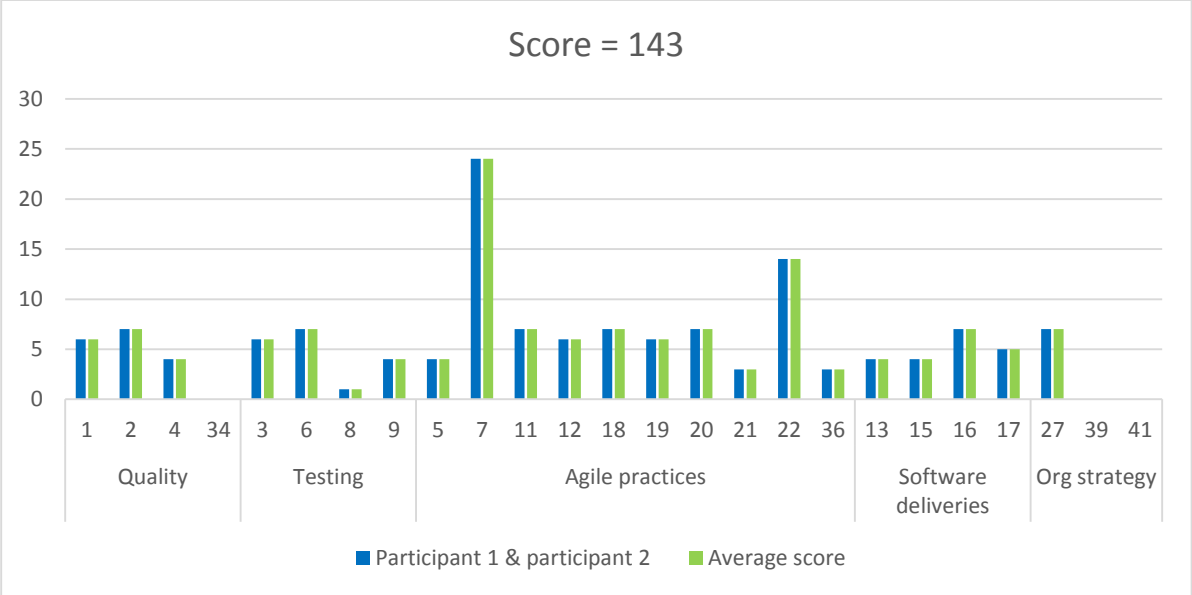

Figure 9: Organization C

Organization D

The development teams of this organization have been implementing agile methodologies for three years. The first respondent has been working for 4,5 years at the company. The second respondent is working 2,5 years at the company. The results from the surveys and the interviews from the two respondents are displayed in the table below.

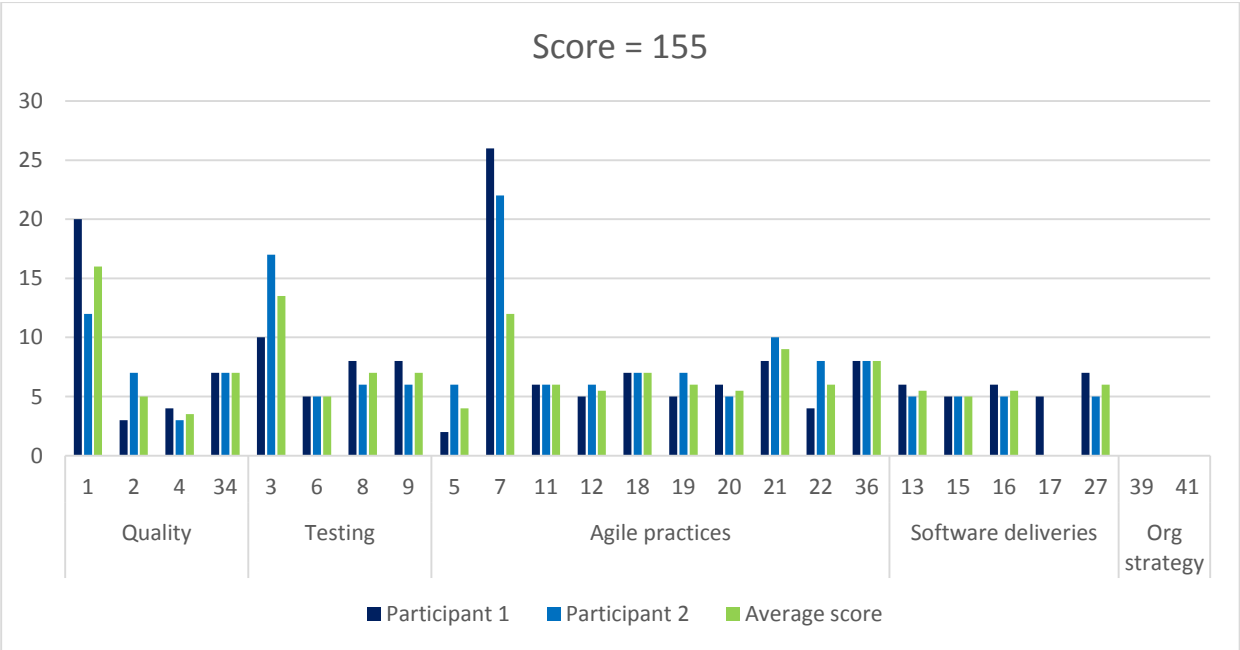

Figure 10: Organization D

Organization E

The participants of organization E have gained their experience at this organization 5 years ago. The company stated implementing Scrum in 2009.

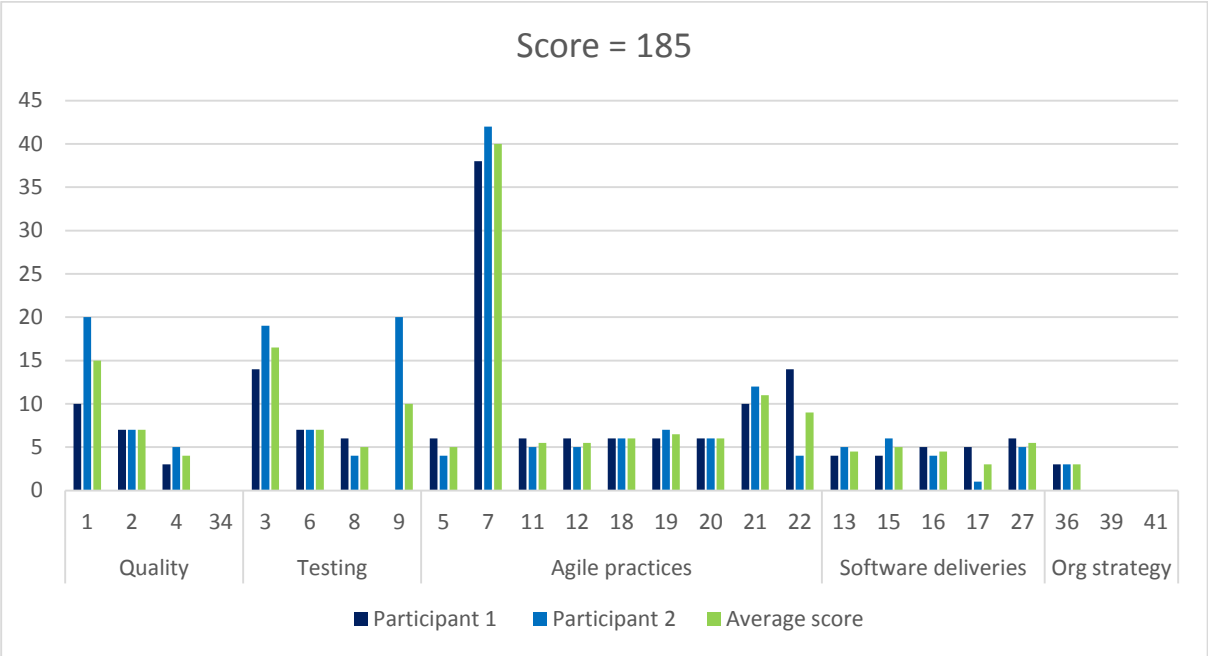

Figure 11: Organization E

Organization F

This company has been started implementing Scrum for three months already. Both participants have started working according to Scrum at this company. They have no previous experience with agile methods.

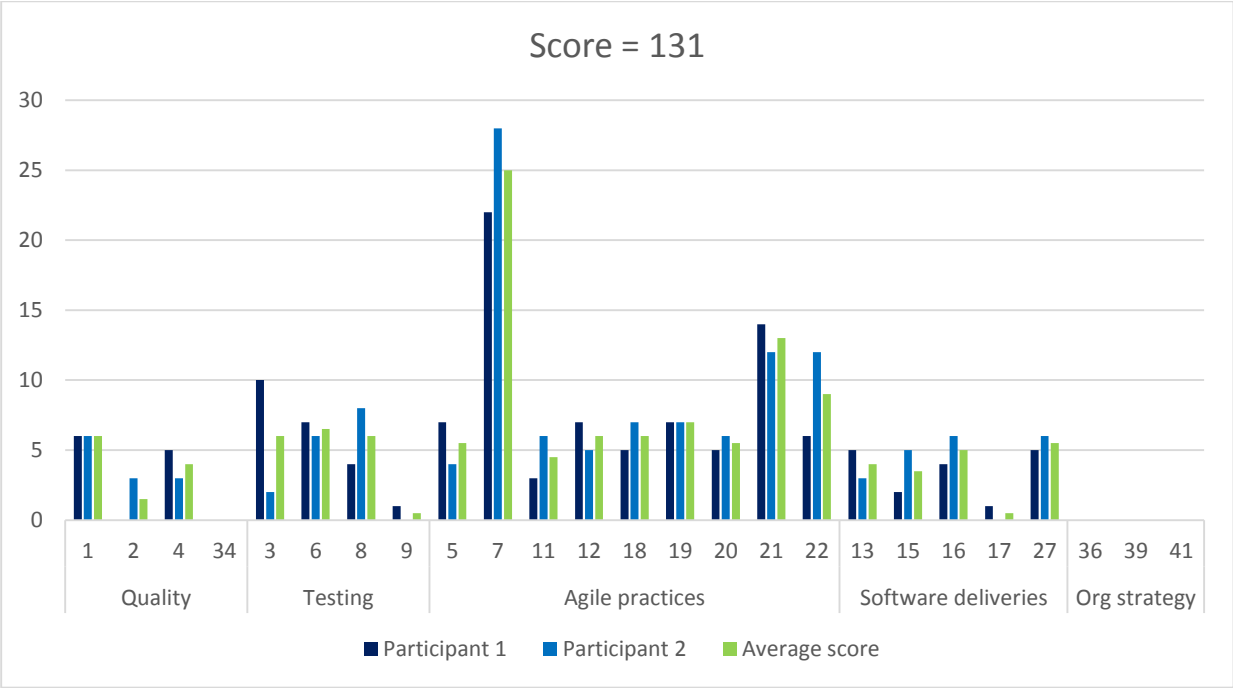

Figure 12: Organization F

Organization G

This organization has been implementing Scrum for 1,5 years. Participant 1 had no prior experience with agile methods and has become familiar with it at this company. Participant 2 has three years of experience with agile methods.

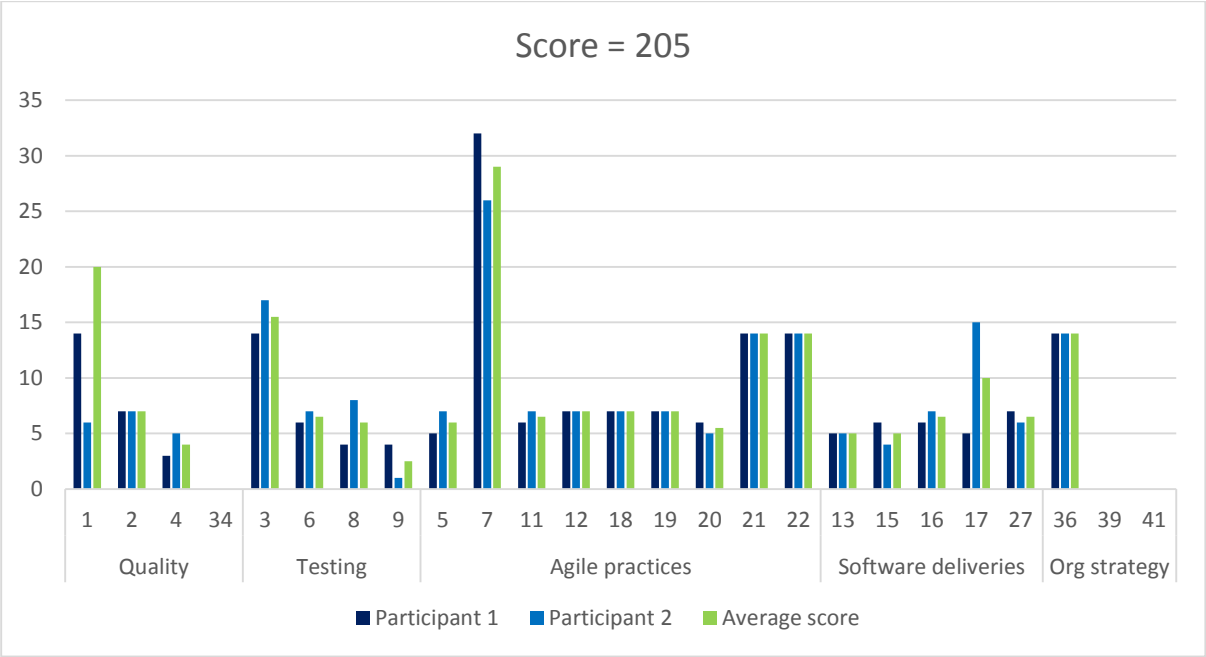

Figure 13: Organization G

Organization H

About a year ago this company started implementing Scrum. Participant 1 has four years of experience with agile methods and participant 2 has been working with agile methods since 2006.

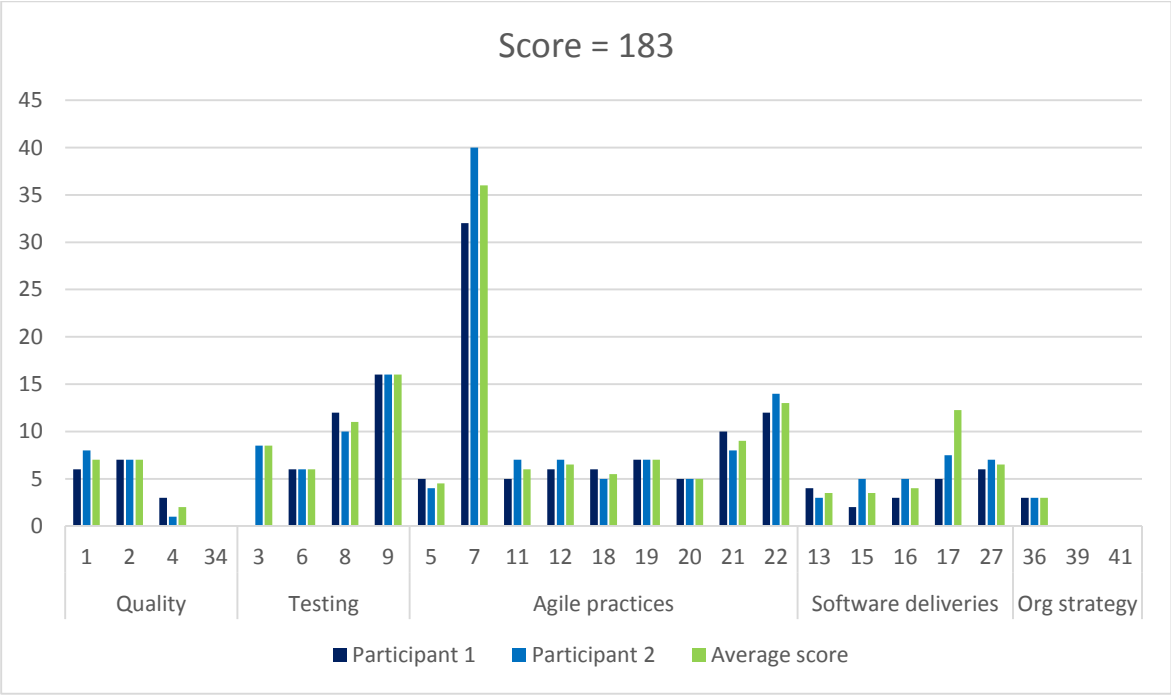

Figure 14: Organization H

Organization I

Participant 1 from this organization has been working according to Scrum principles since 2008. This company has been implementing agile methods since 2008.

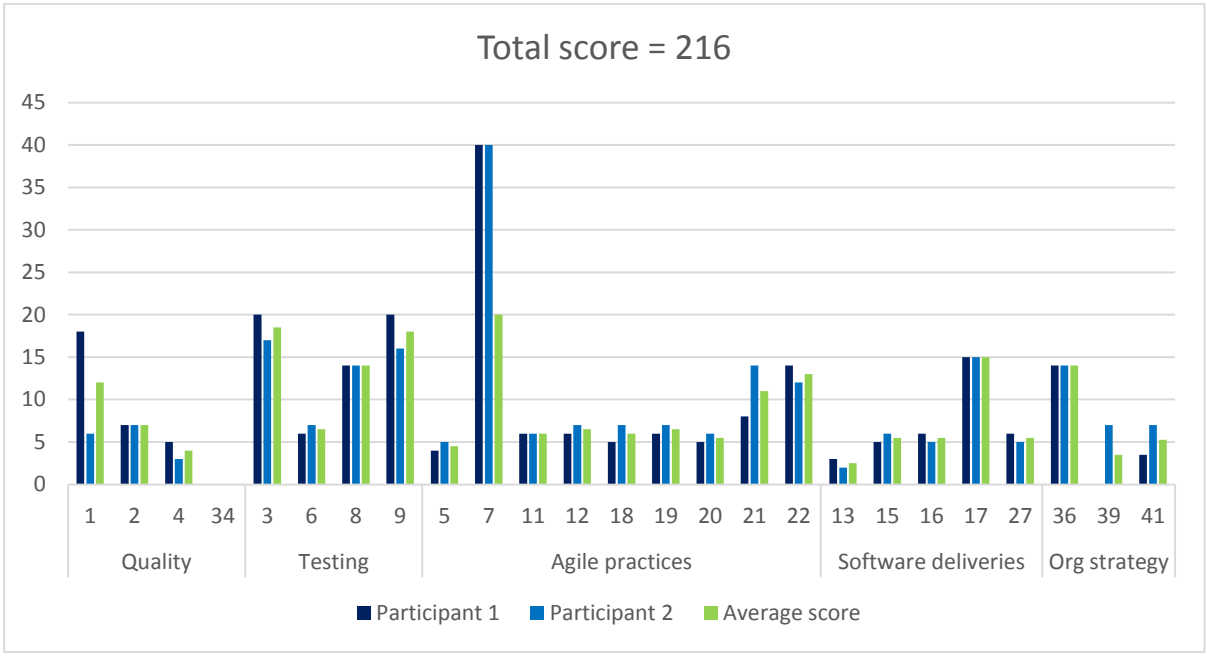

Figure 15: Organization I

Organization J

Participant 1 has experience with agile methods since he joined this company 2 years ago. The company has been implementing Scrum for four years already and participant 2 has been part of one of the first Scrum teams in this organization.

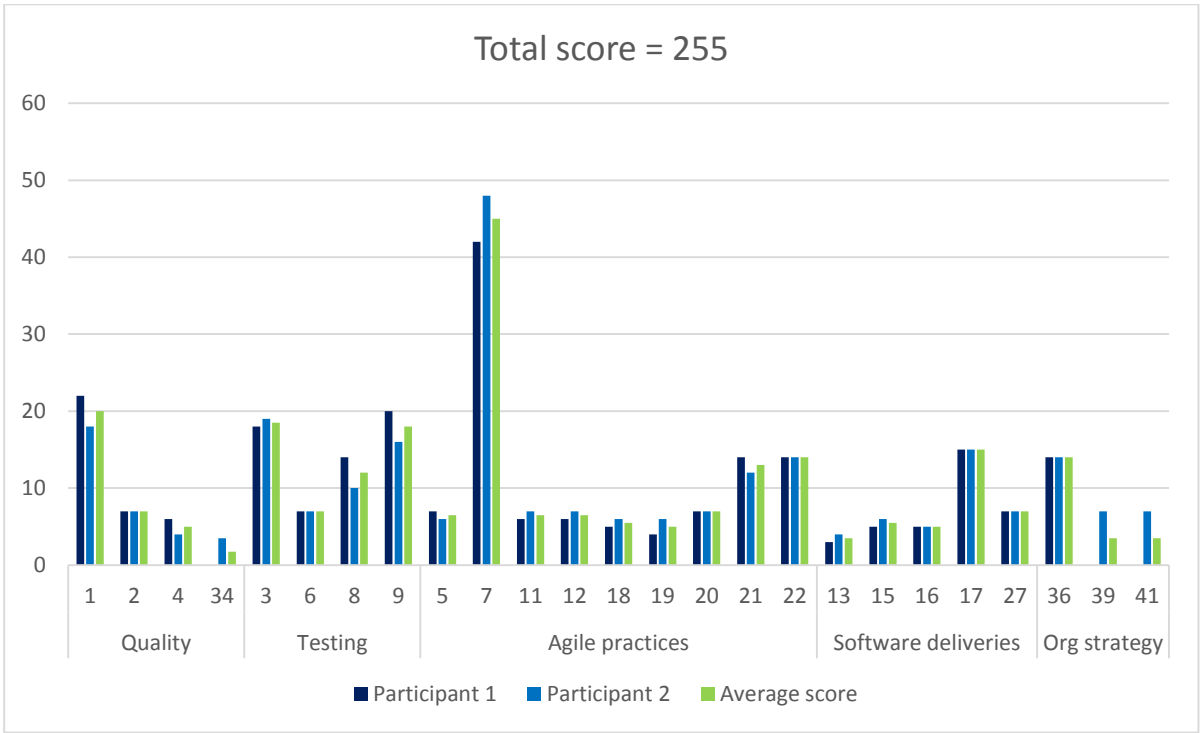

Figure 16: Organization J

Organization K

This organization has been implementing agile methods for eight years already and participant 1 has four years of experience with agile methods.

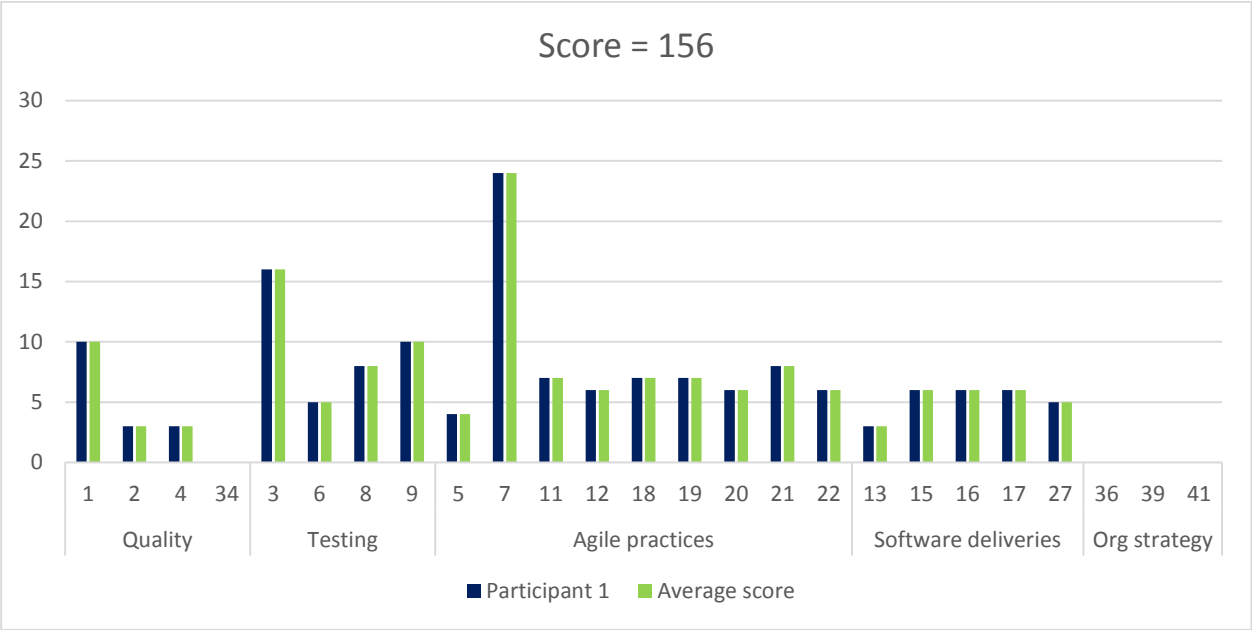

Figure 17: Organization K

4.2.1 Total score

The total score of each organization has been calculated and presented below. The scores are generated by summing up the average score of 25 questions. The maximum score that can be achieved is 301. The results show that organization J has scored most of the points and can be considered the most mature one. Organization J has been implementing agile methodologies since 2011. In contrary, organization F scored the least amount of points and can be considered the least mature. This organization has been implementing agile methodologies for three months.

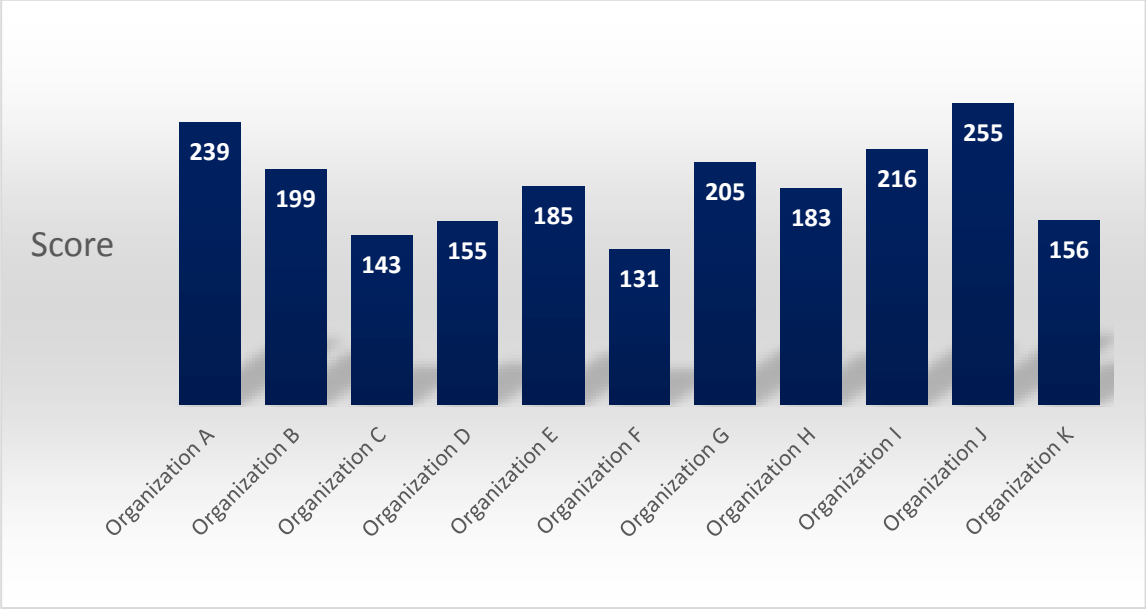

Figure 18: Total maturity score

Based on the results of total score, we distributed the maturity scores into three maturity levels. We have classified the maturity scores in three categories (low, medium and high). The low maturity category consists of organizations that scored less than 150 points. Organizations in this category are considered low mature and are in the beginning stage of agile implementation. The medium maturity level is representing organizations that have implemented agile methodologies for some years, but there is not substantial growth. The score range for the medium level is 150-210. These organizations need to improve on certain areas where they scored a lower score to become high mature. The specific area to improve is presented in section 4.5.5. The high maturity level consists of organizations with a score higher than 210. As a result, we classified the studied organizations in table 14 with the corresponding maturity level.

Low maturity level	Medium maturity level	High maturity level
Organization F	Organization B	Organization A
Organization C	Organization D	Organization I
	Organization E	Organization J
	Organization G	
	Organization K	

Table 14: Agile maturity table

In the figure below we present the scores of each organization versus the experience in agile methods. The experience accounts since the start of implementation of agile methods in the company. The graph can highlight any correlation between the maturity and agile experience.

Figure 19: Maturity vs experience

Earlier in this section we have evaluated the scores and generated a maturity score for each organization. This maturity score is only accountable for the specific team and not for the entire organization. We have developed an overview of all the organizations with their corresponding

maturity scores. This analysis will be used later in this study to find any correlations between the maturity levels and the use of particular quality metrics.

4.3 Effective quality metrics

Quality metrics are indicators to define quality in agile environment. Organizations use a range of various quality metrics in order to deliver high quality products and services. However, it's not clear what actually the effective quality metrics are in agile environment. In this section we will analyze quality metrics in order to answer the following research question: "*What are the most effective quality metrics (QM) that are being used in agile organizations?".* In order to discover the answer to our research questions, we will analyze the questions that are listed below. The participants are requested to answer these questions, then we will present the results related to respondents.

1. What measures or metrics do you collect? Please also specify all the measures that you take but are not on this list.
2. How do you measure software quality?
3. What are the most important quality metrics? Why these?
4. When is a quality metric effective?

4.3.1 Collected measures/metrics

The following table displays the answers given to the question *"What measures or metrics do you collect"*. On the vertical axis the metrics are shown, and the horizontal axis represents the organization.

Metrics	A	B	C	D	E	F	G	H	I	J	K
Number of failed/succeeded auto test	X	X	X	X	X	X	X	X	X	X	X
Code coverage	X	X	X	X	X		X	X	X	X	X
Unit test coverage	X	X	X	X	X		X	X	X	X	X
Number of open customer problems	X	X	X	X	X	X			X	X	X
Total number of automated test cases	X	X		X	X			X	X	X	
Test case count	X	X		X	X		X	X	X		
Defect count during production	X	X		X	X		X	X		X	
Compile failures and build defects			X	X	X	X			X	X	
Weekly defect arrivals		X		X		X	X	X		X	
Fix response time		X	X	X	X			X		X	
Defect count reported by customer		X		X	X		X			X	
Lines of code (LOC)		X						X		X	
Accuracy of estimates	X			X	X						
Others		Sig							Tics	Cc	

Table 15: Applied metrics in organizations

Abbreviations:

- Sig = Sig meter
- Tics = TIOBE Coding Standard Framework (TICS)
- CC = Code Complexity

In the next phase, we classified the quality metrics in number of occurrences, the results are shown in figure 20. A metric can have a maximum occurrence of 11 times. In such a case, the metric is used

in all organizations that are part of this study. The vertical axis is the total number of organizations and the horizontal axis displays the metrics.

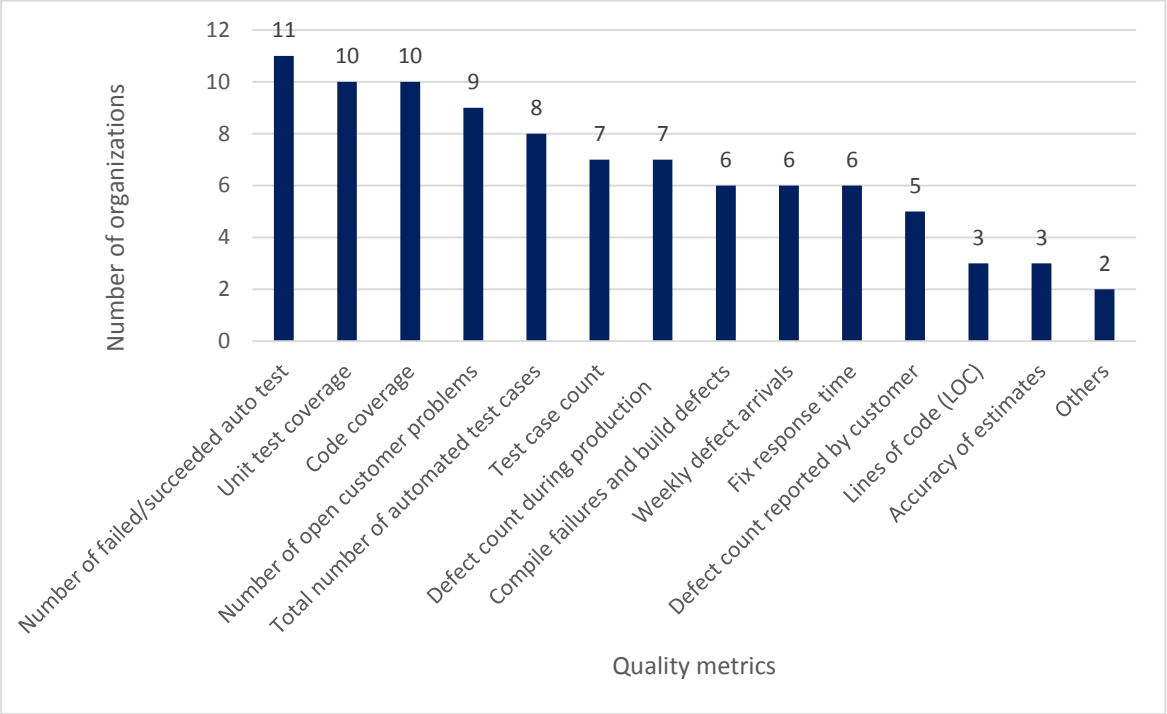

Figure 20: Occurrence of metrics

Based on the results above, we acknowledge that there are four metrics that have the highest number of occurrence and are the most popular ones. These metrics are:

- Code coverage
- Unit test coverage
- Number of failed/succeeded automated tests
- Number of open customer problems

4.3.2 Measure software quality

In figure 21 the responses are displayed regarding the question: *"How do you measure software quality?"* In 53% of the cases, organizations measure software quality using data that is generated by tools. In 41% of the cases organizations use data that is generated manually. And in one case, the data is collected but nothing is done with it.

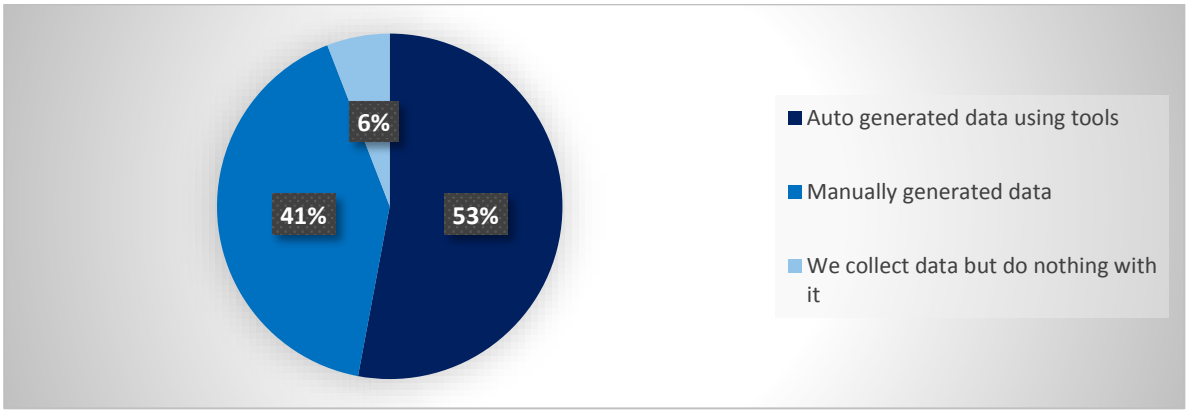

Figure 21: How is software quality measured

4.3.3 Most important quality metrics

The following question to be answered is: *"What are the most important quality metrics?"* The data analyses based on this question concluded the results shown in the table below. The quality metrics that are found the most important for every organization, are selected and linked to the corresponding organization. In some cases, an organization had several choices as the most important quality metric, therefore the extra metrics are also added to the table.

Organization	Most important quality metrics
A	Critical issues (defects)
B	Code coverage
C	Defects
D	Code coverage, defects, test coverage
E	Code coverage, test coverage, code complexity and defects
F	Defects
G	Code coverage, code review, performance test, critical defects, customer feedback
H	Customer satisfaction, defects, acceptance test, unit test
I	Unit test, test case count
J	Unit test, code complexity, defects, secure code scan
K	Defects

Table 16: Most important quality metrics

The collected quality metrics have been further analyzed in order to discover the number of occurrence and popularity. In figure 22 the data regarding these findings are displayed. Nine organizations have found defects or critical issues as the most important quality metric. Four organizations find code coverage as the most important quality metric. The quality metrics with the lowest number of occurrence are: performance test, test case count, acceptance test and code review.

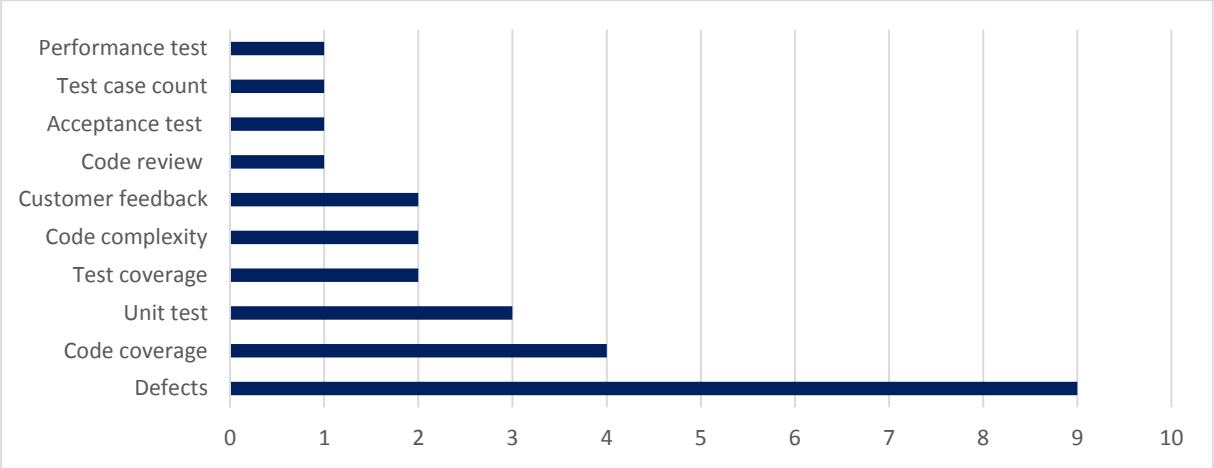

Figure 22: Most important quality metrics

In order to discover the relation between most important quality metrics according to the organizations and the corresponding maturity level of those organizations, an overview is created in figure 23. In this graph, organizations A-K are displayed on the horizontal axis with their most important quality metrics. Some of the organizations had multiple choices in order to choose the most important quality metrics, therefore in some of the cases an organization has multiple quality metrics.

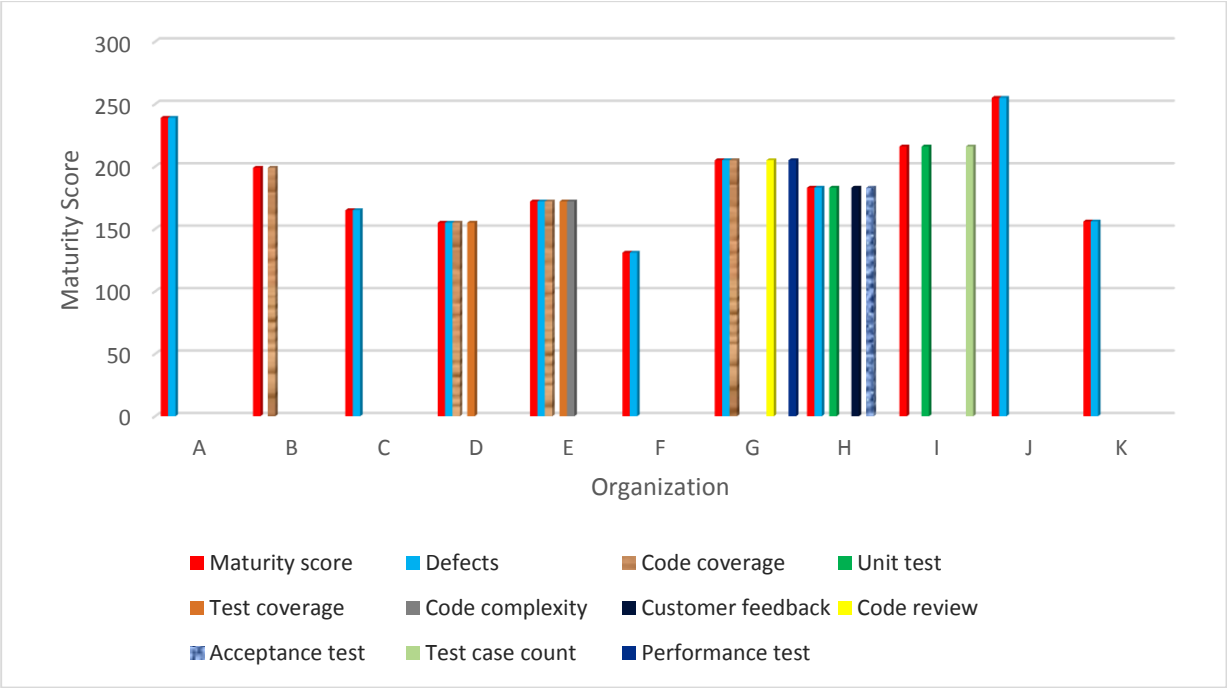

Figure 23: Maturity vs most important quality metrics

4.3.4 Effective quality metrics

During the interview sessions, the participants were asked: "When is a quality metric effective?". The responses to this question are subjective and are based on the experience of the specific individuals. The results are displayed in table 17. The objective of this question is to provide characteristics for a quality metric that can be implemented in organizations. In addition, we can evaluate whether the existing quality metrics have these characteristics. As it is evident from the results, the responses vary between the organizations. However, there are characteristics that are mentioned more than once by the participants and can be considered as important characteristics, these are:

- Provides improvement
- Quick insight/overview
- Be (easy)measureable
- Proven it works

Organization	When is a quality metric effective?
A	Can be automated and provides 100% guarantee (assurance)
B	Provides improvement
C	N/A
D	Should be useful, provides overview and improvements
E	Fast results and its proven that it works, quick insight, engage to improve
F	Detailed overview, quick insight regarding what's wrong
G	Provides a range(good-bad), be measureable, easily obtained, quick insight
H	Smart, related to non-functional
I	Can be compared with baseline
J	Proven it works, easy to measure, must show weaknesses
K	If customer satisfaction is high

Table 17: Effective quality metrics

4.3.5 Outcome

In this section we focus to answer the formulated research question: "*What are the most effective quality metrics (QM) that are being used in agile organizations?".* To answer this, we have analyzed effective quality metrics according to the following questions representing different dimensions:

1. What metrics are used in organizations?
2. How is software quality measured?
3. What are the most important quality metrics?
4. When is a quality metric effective?

Based on the data collected from 11 organizations, there is a variety of metrics collected by organizations. We have selected the top four popular metrics that are being implemented in more than nine organizations. These metrics are:

- Code coverage
- Unit test coverage
- Number of failed/succeeded automated tests
- Number of open customer problems

In the next phase, we analyzed how these metrics are collected with regards to measuring software quality. In almost all of the organizations the data is collected using the two combinations of automatically and manually generating data. In only once case the data was ignored and it was not further processed or turned into something useful.

Subsequently, we needed to identify what are the most important quality metrics according to the organizations. The results derived from this analysis concluded in the following three quality metrics:

- Defects
- Code coverage
- Unit test (coverage)

Finally, we focused on fourth dimension that will identify the characteristics of effective quality metrics. During the interview sessions, experts were requested to provide their opinion related to this question. Many characteristics have been derived from this analysis with a wide range of variety. The four popular responses are presented below:

- Provides improvement
- Quick insight/overview
- Be (easy)measureable
- Proven it works

According to the results in table 18, we can conclude that the popular quality metrics that are currently implemented in organizations, can be considered effective. These metrics match three out of four characteristics for a quality metric in order to be effective. These characteristics are: provides improvement, quick insight/overview and be measurable. When we compare the current metrics with the most important metrics, we can acknowledge that there is a match of 75% in total. In general we can conclude that the most effective quality metrics are code coverage, unit test coverage, defects and the number of failed/succeeded automated tests.

Current quality metrics used	Most important quality metrics
Code coverage	Code coverage
Unit test coverage	Unit test coverage
# Open customer problems (defects)	Defects
# Failed/succeeded automated test	

Table 18: Current quality metrics vs most important quality metrics

4.4 Effective agile practices to support maturity

Agile methodologies consist of various techniques and frameworks. The data outcome of the study demonstrates results from 11 organizations have been studied, all of these organizations are implementing scrum. In this section, the study will target agile practices that are necessary in order to determine agile maturity of an organization. This study will only determine the agile maturity level of a certain team within that organization. The goal of this section is to analyze and evaluate the data in order to identify effective agile practices that can define agile maturity. In order to achieve this, we will focus on these three aspects.

- Which agile practices are applied
- Which agile practices can determine maturity
- Which agile practices are necessary

4.4.1 Applied practices in organizations

In table 19 we present the results with regard to applied agile practices in studied organizations. On the horizontal axis, we have defined the organizations from A-K and on the vertical axis the list of agile practices is displayed.

Practices	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**Total**
Automated builds	X	X	X	X	X	X	X	X	X	X	X	11
Daily standup	X	X	X	X	X	X	X	X	X	X	X	11
Coding standards	X	X	X	X	X	X	X	X	X	X	X	11
Release planning	X	X	X	X	X	X	X	X	X	X	X	11
Continuous integration	X	X	X	X	X	X	X	X	X	X	X	11
Unit testing	X	X	X	X	X		X	X	X	X	X	10
Digital task board	X	X	X	X	X	X	X	X	X	X		10
Refactoring	X	X	X		X	X	X	X	X	X	X	10
Retrospectives	X	X		X	X	X	X	X	X	X	X	10
Scrum poker	X	X		X	X	X		X	X	X	X	9
Open work area	X	X	X		X	X	X	X	X	X		9
Iteration planning	X	X		X	X	X	X	X	X	X		9
Burn down estimation	X	X		X	X		X	X	X	X	X	9
Velocity	X	X		X	X		X	X	X	X	X	9
Dedicated PO	X	X		X	X	X	X	X	X	X		9
Test-driven development	X			X	X	X	X	X	X	X		8
Pair programming	X	X	X		X				X	X	X	7
Collective code ownership	X	X			X			X	X	X		6
Continuous deployment	X	X	X				X		X	X		6
Integrated QA/Dev	X	X			X			X	X	X		6
Automated acceptance test	X	X		X			X		X	X		6
Story mapping	X	X				X			X	X		5
Kanban	X		X	X	X			X				5
Customer acceptance test	X	X						X	X	X		5
Agile games	X			X				X				3
Cycle time							X			X		2

Table 19: Agile practices applied in organizations

As presented in table 19 we can perceive the implementation of agile practices in all organization. In total there are 26 practices that are assessed based on their implementation in agile organizations. In order to construct a more specific analysis based on the popularity of the practices, we summed up the number of occurrence of each practice in each organization and generated the following overview in figure 24. There are five agile practices that are implemented in all studied organizations. These practices are:

- Continuous integration
- Release planning
- Coding standards
- Daily standup
- Automated builds

The top five practices with the least number of occurrences are listed below. The cause of the low occurrence of these practices is not related to their ineffectiveness or unpopularity, in contrary, these practices can be an indication of growth towards higher maturity level.

- Custom acceptance test
- Kanban
- Agile games
- Cycle time
- Story mapping

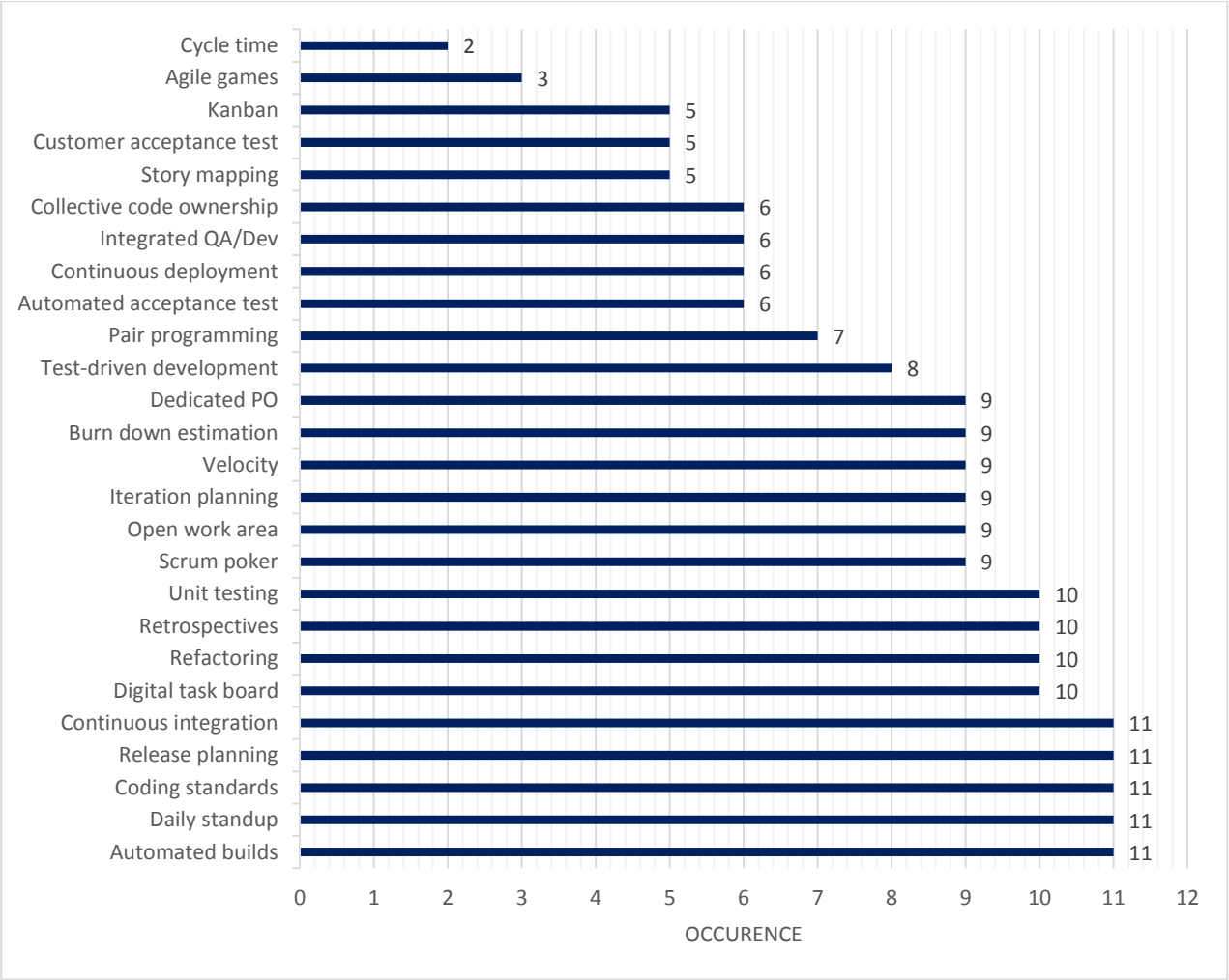

Figure 24: Overview of agile practices occurrence in organizations

4.4.2 Agile practices to determine maturity

The goal of this section is to construct a model that evaluates agile practices in order to determine agile maturity and present effective agile practices that can determine maturity. In order to achieve this goal, the following approach will be followed:

1. Select three organizations with three different maturity levels:
    a. Highest maturity score (Organization J, 255 points)
    b. Medium maturity score (Organization E, 185 points)
    c. Lowest maturity score (Organization F, 131 points)
2. Demonstrate agile practices and their implementations in a table
3. Search for correlation between every agile practice and the three organizations

Practices that belong to **low** level maturity
Practices that belong to **medium** level maturity
Practices that belong to **high** level maturity

Table 20: legend

Organization J (score = 255)	Organization E (score = 185)	Organization F (score = 131)
Scrum poker	Scrum poker	Scrum poker
Open work area	Open work area	Open work area
Iteration planning	Iteration planning	Iteration planning
Automated builds	Automated builds	Automated builds
Daily standup	Daily standup	Daily standup
Coding standards	Coding standards	Coding standards
Digital task board	Digital task board	Digital task board
Refactoring	Refactoring	Refactoring
Release planning	Release planning	Release planning
Dedicated PO	Dedicated PO	Dedicated PO
Continuous integration	Continuous integration	Continuous integration
Test-driven development	Test-driven development	Test-driven development
Retrospectives	Retrospectives	Retrospectives
Unit testing	Unit testing	
Pair programming	Pair programming	
Integrated QA/Dev	Integrated QA/Dev	
Burn down estimation	Burn down estimation	
Velocity	Velocity	
Collective code ownership	Collective code ownership	
Story mapping		Story mapping
Continuous deployment		
Cycle time		
Automated acceptance test		
Customer acceptance test		
Test automation		

Table 21: Agile practices in organization J, E, F

As aforementioned, for this analysis we have selected three organizations with different maturity levels (high, medium and low). As presented in table 21, there are 13 agile practices that take place in all three organizations. These are shown in green color. There are seven practices that are only implemented in two organizations, these practices are presented in yellow. Finally, there are five agile practices that are only implemented in organization J, this organization has scored the highest maturity score according to our study.

Based on the results in table 21, organization F has to implement six agile practices in order to achieve the medium maturity level. As for the organization E, it that has medium maturity level, this organization has to implement six agile practices in order to promote to a higher maturity level. An overview is presented in table 22 below.

From low to medium level	From medium to high level
Unit testing	Story mapping
Pair programming	Continuous deployment
Integrated QA/Dev	Cycle time
Burn down estimation	Automated acceptance test
Velocity	Customer acceptance test
Collective code ownership	Test automation

Table 22: Agile practices that indicate growth

Based on the results in table 22 we can consider these agile practices to be necessary in order to grow in maturity. On the other hand, we can see similarities in the use of agile practices in the lower level of maturity. As already mentioned, these similarities are based on the implementation of 13

agile practices that take place in all three organizations. In the medium level we acknowledge the similarities in implementation of agile practices between two organizations (organization J and E). This implies that, organizations with medium level are implementing the practices that organization J and organization E have in common.

4.4.3 Necessary agile practices

In order to identify effective agile practices, we investigate the necessary agile practices in organizations. The participants are experts with knowledge and practical experience in the area of agile implementation. They have been requested to indicate the necessary agile practices, according to their beliefs and experience. The list of necessary agile practices is a strong indication for determining the first level of agile maturity. These practices are seen as essential practices that will differentiate agile organizations from non-agile. In table 23 we have listed all organizations with their corresponding response on necessary agile practices.

Organization	Necessary agile practices
A	Daily stand-up, retrospective
B	Retrospective, backlog refinement, openwork area, demo, continuous deployment
C	N/A
D	Daily stand-up, retrospective, code review, planning
E	Daily stand-up, retrospective, planning, backlog refinement, burn-down chart
F	Daily stand-up, retrospective, planning, openwork area
G	Daily stand-up, retrospective, code review, planning, burn-down chart
H	Daily stand-up, retrospective, backlog refinement
I	Daily stand-up, planning, openwork area, pair programming
J	Daily stand-up, retrospective, backlog refinement
K	Retrospective, planning

Table 23: Necessary agile practices in organizations

To gain a more detailed analysis of necessary agile practices based on their implementation and popularity, we have presented these in figure 25. This figure presents all necessary agile practices, according to experts and agile practitioners based on the number of occurrences. It is evident that retrospective has the highest popularity, followed by daily stand-up and the planning session.

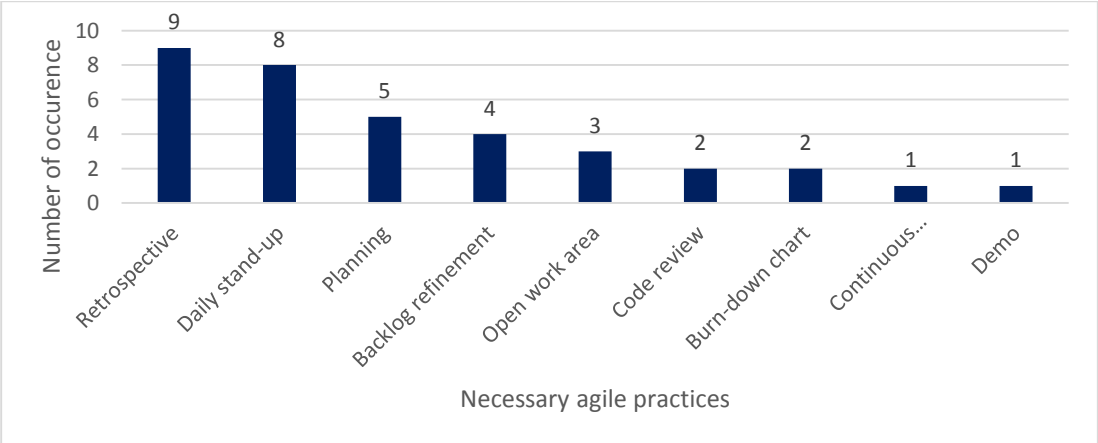

Figure 25: Agile practices that are considered necessary

4.4.4 Extra elements

In addition to the existing agile practices that mostly identify organizations with lower maturity, we pay attention to the agile practices that are generally considered to be residing in the higher maturity levels. We specifically analyzed the implementation of test automation and continuous delivery in organizations. The results show that, the top three organizations with the highest maturity score have successfully implemented continuous delivery and test automation. On the test automation subject, we evaluate two aspects: 1) test automation implementation, 2) test automation coverage. Organizations with medium level of maturity have started implementing continuous delivery and test automation. And organizations with the lowest maturity score are not performing any activities related to both aspects.

The extra elements are specifically evaluated in detail for organizations J, E and F which represent the three maturity levels.

Organization	Continuous delivery	Test automation	Test automation coverage
J	Yes	Yes	90%
E	Started	Started	5%
F	No	No	0%

Table 24: Implementation of continuous delivery and test automation

4.4.5 Outcome

In section 4.4 we analyzed results in order to discover effective agile practices that support maturity. In this section we explore the findings and provide a short summary. Subsequently, we provide the suggested answer for RQ2: *"What are the most effective agile practices that support maturity?".*

Initially we investigated the applied agile practices in organizations, identified agile practices that can support maturity, then listed all necessary agile practices and finally evaluated the extra elements. We combined the results of four aforementioned subjects and mapped them below in table 25.

Top 5 popular practices	5 least popular agile practices	Agile practices that can determine maturity	Necessary agile practices	Extra elements
Continuous integration	Agile games	Unit testing	Retrospective	
Daily stand-up	Kanban	Pair programming	Daily stand-up	
Coding standards		Integrated QA/Dev	Planning	
Release planning		Burn-down estimation	Burn-down estimation	
Automated builds		Velocity	Backlog refinement	
		Collective code ownership	Open work area	
	Story mapping	Story mapping	Code review	
		Continuous deployment	Continuous deployment	Continuous deployment
	Cycle time	Cycle time	Demo	
		Automated acceptance test		
	Customer acceptance test	Customer acceptance test		
		Test automation		Test automation

Table 25: Summary of four dimensions of agile practices

Agile practices that appear more than once are shown in the same color. There are in total seven agile practices that are mentioned in more than one subject. These practices are considered as effective agile practices to support maturity.

Based on the results in table 22 and table 25, the answer to our research question of this section is described in table 26. These practices are categorized into three levels to provide a better overview.

Low level	Medium level	High level
Daily stand-up Burn down estimation Integrated QA/Dev	Unit testing Pair programming Velocity Collective code ownership	Continuous deployment Story mapping Cycle time Automated acceptance test Customer acceptance test Test automation

Table 26: Effective agile practices to support maturity

4.5 How do organizations measure agile maturity

Organizations strive to excel in different areas, especially in agile development. But in order to excel, organizations need to become aware of their performance and development. They need to apply measures and assessments in order to understand what their current position is in agile development and what the improvement areas are. To perform this, organizations should have the commitment to improve and focus on agile excellence. Therefore, in this section we focus how organizations assess and measure agile. To compute the related results, different dimension of the topic has been investigated in order to develop a more in-depth analysis. The following dimensions are considered:

- Agile assessments
- Agile training/workshops
- Agile experience of the team
- Agile certifications
- Improvement areas for teams
- Missing agile practices
- Organizational focus points to improve on agile

4.5.1 Agile assessment

Information related to the current state of implementation of agile developments provides organizations awareness with regards to agile growth. The question remains whether the organizations are interested in such a data and how much effort is invested to acquire it. This section describes data related to agile assessment in 11 organizations.

The studied organizations have been asked whether they have executed any form of agile assessment in order to get knowledge related to their current state of agile implementations. Any form as assessment is acceptable if it provides knowledge of the current state of agile implementation. We will not differentiate between in-house assessments or external. In-house assessment is conducted by the company itself using their own staff or experts, and external assessment is conducted by an external party, this could be a company that is specialized in assessing organizations on agile.

In figure 26 we present the studied organizations and their approach on agile assessment. As it's evident from this graph, only four organizations are conducting agile assessments.

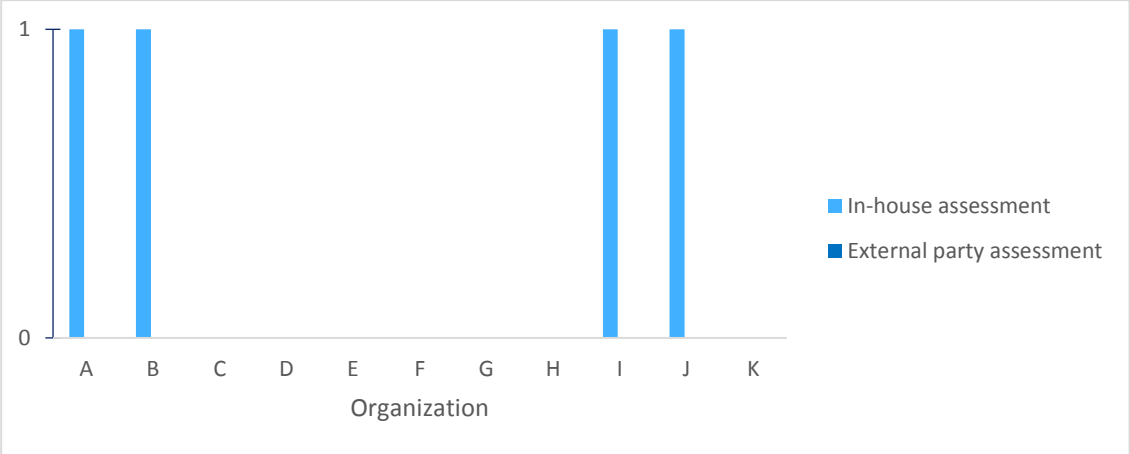

Figure 26: Assessment in organizations

Organization A is according to our study considered to be an organization with a high maturity level. This organization scored 239 points from the agile maturity test. This organization is mainly performing in-house assessment by their own agile coaches. These assessments are conducted on a regular basis and are based certain models used, one of these models is the Tuckman model (Bruce W. Tuckman, 1977). The teams are assessed according to the four stages of the model described below and then it's decided how to improve these teams.

1. Forming
2. Storming
3. Norming
4. Performing

Organization B scored test 199 points and is considered to have a medium level maturity. This organization is performing agile assessments based on the new projects and products. Agile experts and their own staff are responsible for conducting these assessments.

Organization I is considered to possess a high maturity level with 215 points. This organization conducts assessments from time to time by own staff and agile experts active in this company.

Organization J is also considered to have a high maturity level with 255 points. The assessments are conducted on a regularly base by own staff and agile experts working in this company.

We can conclude that in general, organizations with a low and medium agile maturity level are not focusing on agile assessments. They are mainly occupied implementing new agile practices and improving the existing ones. However, organizations that are considered mature in agile, are conducting assessments in order to continuously improve. These assessments are conducted by sending out surveys and meetings with the teams. In organization A, teams are sometimes during the retrospective sessions assessed.

4.5.2 Agile training/workshops

Training and workshops can be essential to improve in agile implementations. Based on the investigations conducted in 11 organizations, it is evident that team maturity plays an important role. Therefore, some of the teams might need more assistance to improve on certain areas. Workshops and trainings are the solutions to increase the team maturity. Agile training and workshops can also be implemented to increase the level of expertise and knowledge of individuals within the organization. In this section we present results with regard to agile training and workshops in organizations.

Nearly all investigated organizations have provided training and workshops in the beginning when agile development was introduced. We will not account these trainings and workshops as a valid result of our research question. We only focus on training and workshops that are provided afterwards to improve agile development in a later phase. The training and workshops in the initial state of implementation are excluded from the results.

As depicted in figure 27, agile training and workshops are neglected or not applied in 64% of the cases. Only 36% of the organizations are providing training or workshops to improve agile development. These trainings and workshops are provided on a regular basis. Organizations that are providing these services are:

- Organization A
- Organization B
- Organization I

- Organization J

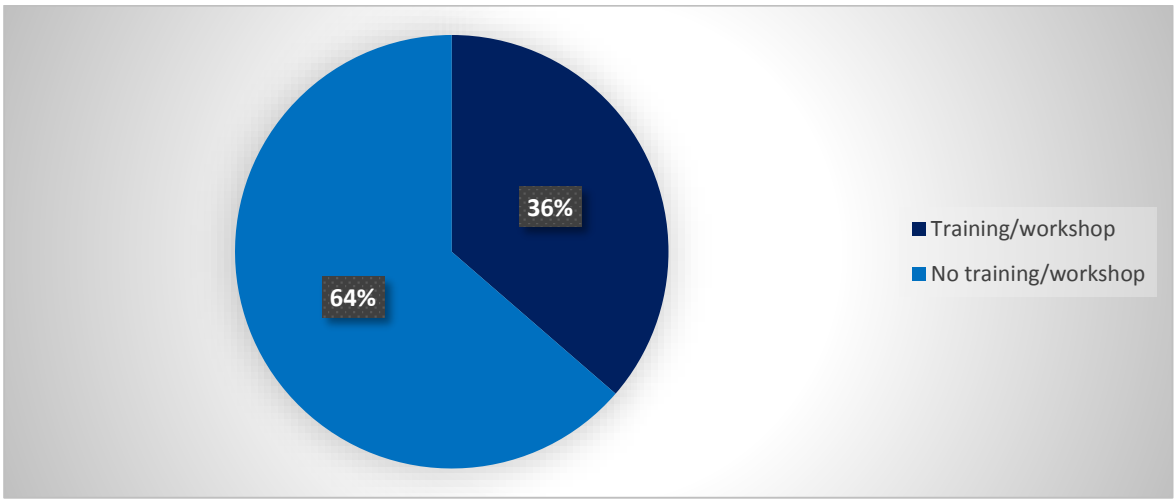

Figure 27: Training and workshop

From the four organizations that do provide agile training and workshops, three of them are considered to have a high maturity level according to our assessment, the other one has medium maturity level.

Based on the results, only organization A is providing extensive agile training for the teams by agile coaches. There are 36 scrum teams in this organization with different maturity levels. On a regular basis, the teams that are struggling with certain agile practices are put under scrutiny. Subsequently, these teams receive the necessary knowledge and training to achieve the desired level.

Organization B provides trainings on a regular basis and is mainly implemented on the kick-off of new projects. These trainings are provided by agile coaches and the focus point is mainly SAFE implementation.

The trainings provided in organizations I and J are mainly meant for product owners and scrum masters. Other team members are not involved in these trainings.

The results derived from this section concludes that agile training and workshops are provided by nearly all organizations when agile development is introduced. At a later stage when agile development has been implemented, trainings and workshops are not provided in low and medium mature organizations. Although some of medium mature organizations provide trainings, these are only provided for scrum masters and product owners. The results show that not all the mature organizations provide training or workshop for the team, only in some of the cases it occurs.

4.5.3 Agile experience of the team

In this study, we analyzed the agile experience of representing teams. The experience of every team member of each team is considered to calculate the total experience of the team. The participants have indicated their team member's experience in order to define the total team experience working according to agile principles. In some cases, she participants have indicated that they don't know the exact experience of their team members, but they can provide a general estimation what the experience of their team members is. Below we describe how experienced the teams are working with agile development. We evaluate the experience of the teams in the following categories:

- Poor – on average the team has few months experience
- Fair – on average the team has 1+ year experience

- Good – on average the team has 2+ year experience

The team experience from the corresponding organization is described in table 29.

Organization	Team experience in agile
A	Good
B	Good
C	Poor
D	Fair
E	Good
F	Poor
G	Fair
H	Fair
I	Good
J	Good
K	Fair

Table 27: Team member's experience with agile

Teams with "good" experience are derived organization A, B, E, I and J. Three out of these five organizations have already scored a high maturity score in this study, these are organization A, I and J. Maturity scores can be found in section 3.3.1. Teams that have "fair" experience in agile, are representing organizations with a medium maturity level. Organization C and F have "poor" team experience and according to our maturity test, these organizations scored a low maturity score.

4.5.4 Agile certifications

The study investigated the level of expertise of agile teams based on agile certifications. The teams were asked which of the team members has any form of agile certifications. The results are presented in figure 28.

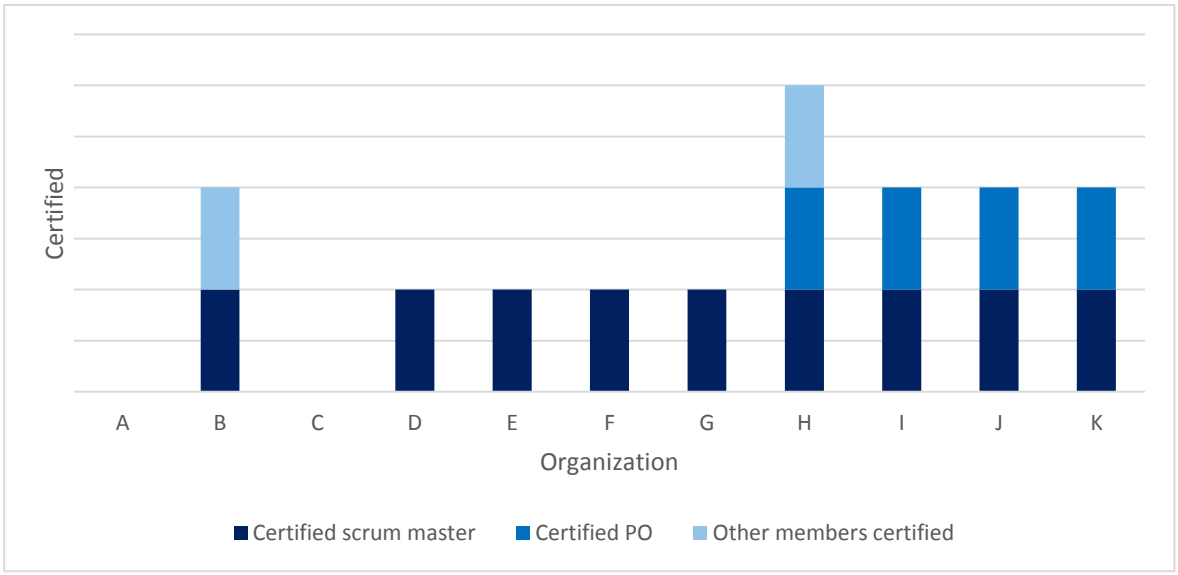

Figure 28: Agile certifications in organizations

Nearly all organizations have a certified scrum master except organizations A and C. Organization A is considered as a high level agile maturity. The scrum masters in this organization are trained in house by agile coaches. There are no formal agile certifications for these scrum masters. According to agile

coaches of this organization, certification does not improve the capabilities and performance of scrum masters. In addition, agile coaches believe that the training provided by them has a significantly more quality and is tailor made than the training provided by an external company. As for organization C, it's a start-up company and there is no emphasis on following practices by the book. The scrum master has developed his knowledge individually be self-learning.

4.5.5 Improvement areas for teams

Implementing agile methods are typically associated with hurdles that teams undergo, even the most mature teams could have small issues to deal with. The teams have to tackle these struggles in order to resolve it. Improvement areas are typically derived from the issues that teams have in implementing agile methodologies. These improvement areas could be an indicator to discover the difficult practices in agile.

The teams studied in this research have highlighted their possible improvement areas in order to perform better. The representing teams of each organization are listed below in figure 29 with the improvement areas that have been pointed out by the teams self. The list of improvement areas linked to each organization can be found in the appendix section A.

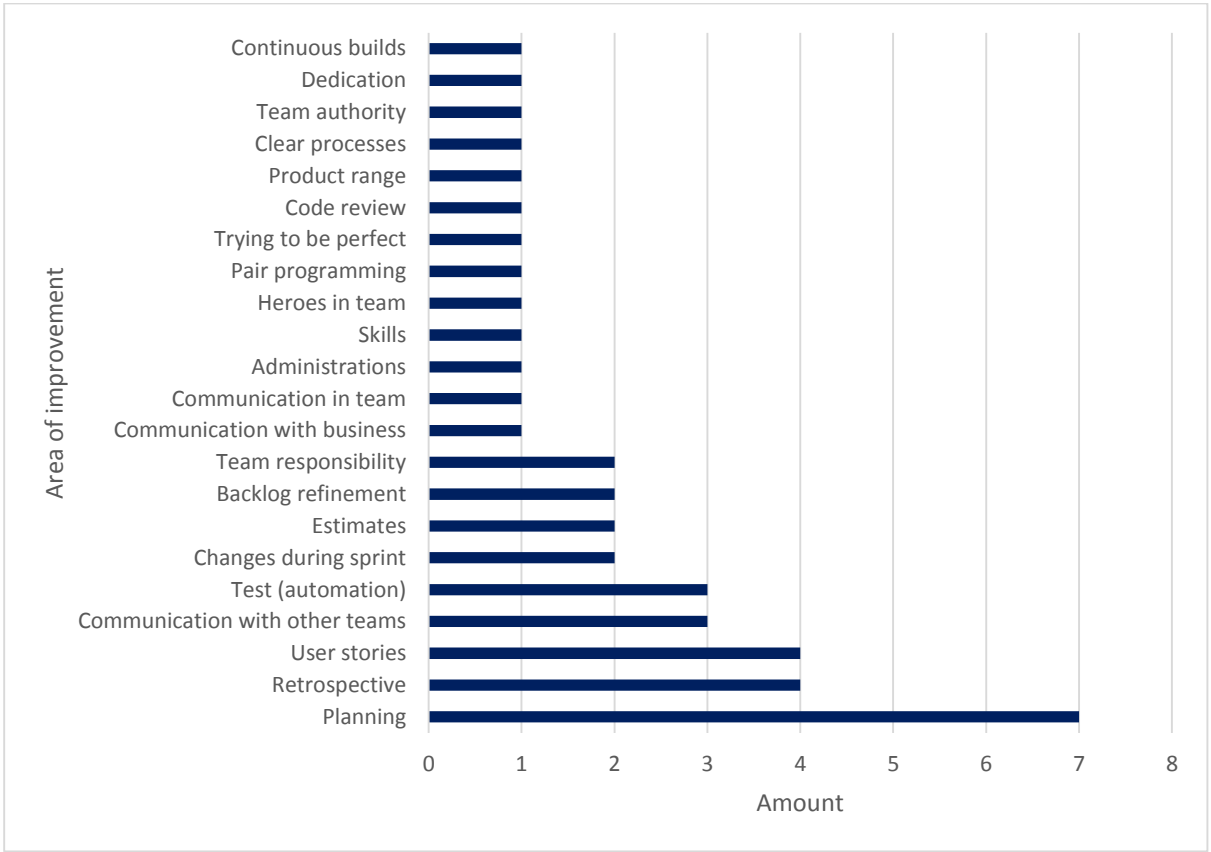

Figure 29: Improvement areas indicated by teams

As described in figure 29, the most mentioned improvement is planning. The data analysis has shown that teams find it hard to perform a good sustainable planning session for their iterations. The issues related to planning sessions are often:

- Planning takes too long
- Planned too tight
- Cannot plan certain issues (bugs)
- User stories are not defined properly, cannot properly plan

The second most mentioned improvement areas are users stories and retrospective. The teams have indicated that user stories are often incomplete or not well defined. As a result, it's hard for the team to understand the user story and create sensible tasks. Consequently the sprint is started with those user stories and the team is struggling to understand and adjust the user story during the sprint. This effect will sometimes result in a failed sprint due to not meeting the vague requirements.

As for the retrospectives, four out of 11 teams have indicated the necessity to improve on this. Opening up and following up on action points are often the issues that are related to retrospectives. In some cases, it takes some time for the team to express their underlying feelings about the sprint and really contribute to retrospectives. In other cases, the retrospective sessions are successfully conducted and useful issues have been highlighted. However, after the retrospective, the created action points are often neglected and forgotten.

4.5.6 Missing agile practices

Studied organizations often offer teams and individuals the freedom to implement agile methodologies. Although there is freedom, sometimes there are some agile practices that teams would like to do but these are lacking. We will focus on the team's view and highlight the missing agile practices according to the teams studied.

Organization	Missing practices according to team
A	None
B	Devops, continuous delivery
C	N/A
D	Pair programming, test driven development
E	Continuous delivery, test automation
F	Code review, unit test, code quality tools, continuous builds, integration tests
G	Pair programming
H	Test automation, acceptance criteria, risk analysis
I	None
J	None
K	Metrics, velocity

Table 28: Desired agile practices that are not in place

Based on the results in table 30, three teams representing from organization A, I and J have indicated that there are no agile practices or techniques that they would like to conduct that are not in place. Furthermore, continuous delivery, pair programming and test automation are agile practices that are missing in some organizations.

4.5.7 Organizational focus points to improve on agile

In the previous section we presented results with regards to preferences of agile teams and agile practices that they would like conduct that are not in place. The next phase is to analyze the organizational point of view on agile development and discover the possible correlations. Specifically, what are the focus points of organizations to improve agile development? Some organizations are successful in implementing agile in such a way that there is no significant need for improvement. Whereas other organizations have a wish list of certain goals related to agile to achieve.

In table 31, an overview is provided representing organizations and the desired focus point to improve.

Organization	Focus point
A	Autonomy, reduce team dependency
B	Continuous delivery, SAFE
C	Test quality
D	Agile roles, shippable products, empowering teams
E	Continuous delivery, test automation
F	Code production, continuous product improvement
G	Estimates, UI test automation
H	Test quality(risks and automatic acceptance test)
I	-
J	Bi-monthly epics live, bi-weekly small release
K	Introduce QA role, feature teams

Table 29: Organizational focus point to improve

Comparing table 30 and 31 has shown that, there is a small number of correlations between the team's point of view and the organization on agile. Organization B indicated to focus on continuous delivery, the team indicated that continuous delivery is missing or not fully implemented. The same is seen in organizations E, F and H. For these organizations the focus point is related to continuous delivery, continuous improvement and test quality. As for the other organizations, we can conclude that there are no correlation or similarities. For the high performing organizations or the ones with high maturity, we assume this doesn't indicate a negative view, however, for the low maturity organizations, this could have a negative effect as it is related to slower growth in maturity. For the low maturity organizations this could have various unknown reasons, the assumption is that the message is not well communicated to the teams, or the commitment of the team is lacking.

4.5.8 Outcome

In sections 3.6.0 to 3.6.7 we analyzed dimension related to how organizations measure agile development. Now we provide a short summary with a suggested answer for RQ3: "*How do organizations measure the maturity of their agile software development?*"

Generally, most organizations that implement agile methods don't perform any assessments within the company to understand their level of agile implementation. However, some of the mature organizations do assess teams in order to continuously improve. Typically, every studied organization has indicated that training and workshops are provided in the initial stage of agile implementation. In the later stage, only a selected number of mature organizations are providing training and workshops to improve teams.

We studied the experience of individuals within an agile team. It is evident that team formation in mature agile organizations is consisted of people with fair to good experience with agile methods. The experience with agile methods in organizations with lower maturity are in most of the cases very little and considered poor. Nearly all organizations have a certified scrum master except for two organizations.

The results indicate that agile certifications are not a guarantee for successful guidance and implementation of agile methods. This specific evidence derives from organization A. This organization is considered very mature, however, there are no certified scrum masters active in this organization. All the scrum masters have been developed and trained in-house by internal agile coaches.

We analyzed the improvement areas suggested by the teams, this indicated that planning is the most difficult practice in seven organizations and certainly needs improvements. Subsequently,

retrospective and defining proper user stories are the second most difficult practices that need improvement.

The next phase was to investigate the missing agile practices and possibly discover correlations with the organizational focus points. This analysis showed that generally the mature agile organizations are not missing agile practices, all the desired agile practices are implemented. The correlation found is related only to low and medium agile mature organizations. The similarities are seen between the missing practices described by the team and the organizational focus points.

Finally to answer how organizations measure agile maturity, we have to evaluate the aforementioned dimensions. Based on the results presented, from the mature organizations, only a few organizations are assessing the teams. The assessment is conducted by sending out surveys, creating meeting with the teams and requesting input from the team during the retrospective meetings. One organization makes use of theoretical models and frameworks in order to measure maturity. Organizations in the range of low and medium agile maturity are not investing any effort in assessing teams on agility. Organizations in the low and medium range of agile maturity are not measuring their maturity.

4.6 Impact of successful or failed projects on quality metrics

In general, agile methodologies are adopted because of delivering fast software delivery and increased the chance of successful projects. However, implementing agile methodologies can lead to failed projects and is not always a guarantee for success. This failure can be based on different aspects, for instance; low skilled individuals, no structural guidance for implementing agile methods and many other aspects. In this section we focus on successful or failed projects and the impact on quality metrics.

First, we need to define what is meant by successful or failed projects. Successful projects is derived from satisfaction, functionalities, but also from quality attributes such as performance, usability and reliability (Jeon, Han, Lee, & Lee, 2011). Successful project stands for a successful delivery of a complete product the way the customer had desired, the customer should be happy. In contrary, failed projects means that the product is not complete and the customer is unhappy about the delivered product. In some cases the projects can be terminated half way due to customer's feedback or the solution is not realistic for the market.

To understand the impact of successful or failed projects on quality metrics, we investigated organizations on the following topics:

- Rate of successful projects
- Customer feedback
    - When received
    - What feedback
    - Impact of feedback
- Measures due to poor quality

4.6.1 Rate of successful projects

Experts and representatives from 11 organizations have been requested to indicate the success rate of the projects in their organizations that they have been involved with. We specifically requested the success rate of the projects that the team has been involved with, the results are not representing the organization as a whole. Using this, we try to find a relation between the rate of success of projects, customer feedback and measures initiated based on failed projects. Customer

feedback is described in section 3.7.2, and in section 3.7.3 we find the results based on measures taken due to failed quality.

The results of success rate are presented in table 32.

Organization	Rate of successful projects in percentages
A	90
B	73
C	100
D	80
E	70
F	80
G	85
H	90
I	90
J	100
K	90

Table 30: Rate of successful projects in agile organizations

The study clarified that generally participants find it hard to measure project success, these results are based on rough estimations. The results showed that in some cases, organizations are running projects for a long period of time, this varies between one to three years. Organizations that run the same project for a couple of years, have indicated that the success rate remains 100% if there are no failures within the project. An example is organization C, it's a start-up company and is running the same project for two years. The project has been successful until now without any failures, therefore the success rate is 100%.

It is evident that in general the success rate of the projects is relatively high. It is interesting to find out why the success rate is as provided by the participants. Therefore, the representing participants were requested to elaborate why the success rate is as they described. As for the organization with a high success rate, the underlying motive was related to the adoption of agile methods, following short iterations, good collaboration between business and IT and having small teams. Furthermore, code reviews and many aspects of testing were mentioned. Especially the test coverage and automated testing were described as factors related to this success rate. Organizations with the relatively lower success rate, related this number to over-commitment of the teams. As a result the success rate dropped in these organizations.

4.6.2 Customer feedback

Customer feedback is an indicator for software quality. When the customer is happy, the feedback is positive and software quality is high and satisfying. In this section we study organizations specifically on what customer feedback is received, when it's received and what the impact was on development or the testing process.

Organization	What feedback	When	Impact
A	Issues Customer satisfaction	1 month	Listen more to stakeholders Involve customer early in process
B	Customer satisfaction	2 weeks	No
C	Issues usability of product	After sprint release	Only impacts if many customer complain
D	Customer satisfaction	-	More test automation

			Code reviews
E	Issues Usability Performance Documentation	Few months	Change of scope Increase test coverage
F	Delivery status System stability Customer satisfaction	1 week	More time for testing UI design
G	Issues Adding features	1 month	Delivery time Change of scope
H	Issues Functional feedback	After release to couple of weeks	More and broader view on testing Test coverage improvement
I	Issues Customer satisfaction Compliments	1 month	Refactoring More testing
J	Customer satisfaction Adding features	2 months	Faster acceptance test Test coverage Assess our environment
K	Customer satisfaction	1 week	No

Table 31: Customer feedback based on three dimensions

According to the results of table 33, most of the feedback provided by the customers is related to customer satisfaction and the issues reported. Customer satisfaction is often expressed by indicating whether the release was good or bad. In some cases, organizations receive feedback in terms like "it works fine". Other form of feedback is the reported issues by the customer. In case of malfunction or an error, organizations are notified by the customer and request immediate fix. The results show that only the critical issues reported by the customer are seen as a form of feedback. Other forms of feedback provided by the customers are usability of the product and feature requests. Figure 30 presents the aforementioned feedback types and its distribution.

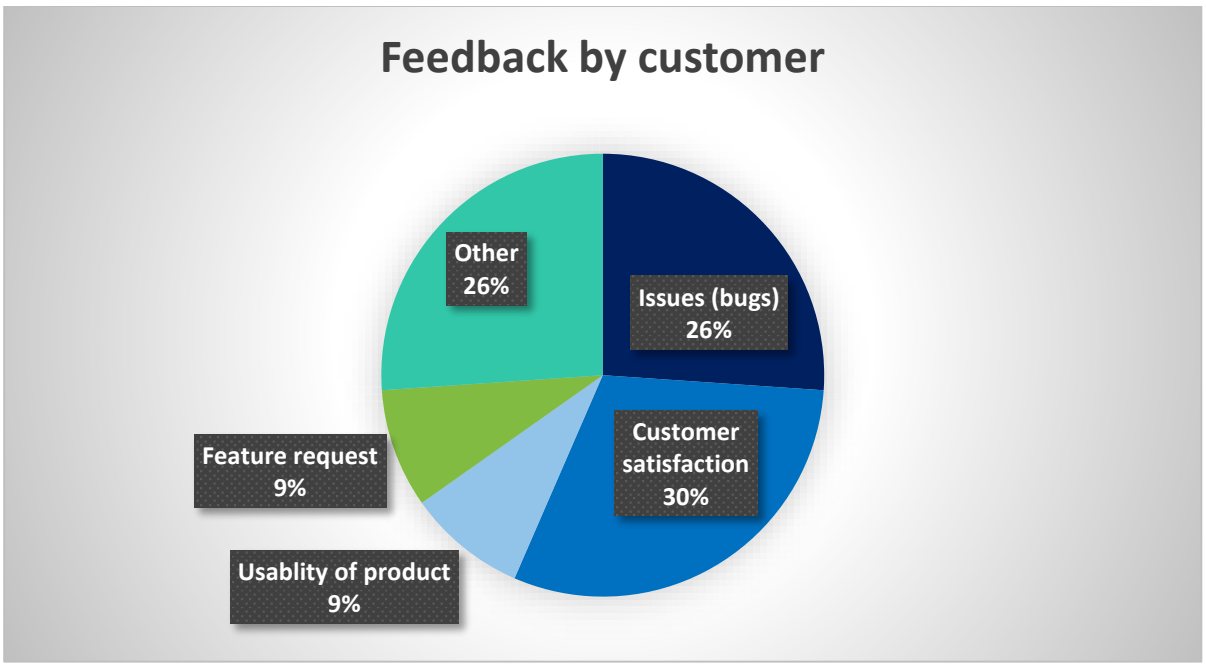

Figure 30: Types of feedback

The feedback provided by the customer can be immediately after the sprint release or it can take months before any feedback is given. Figure 31 displays when the feedback is provided by the customers in studied organizations.

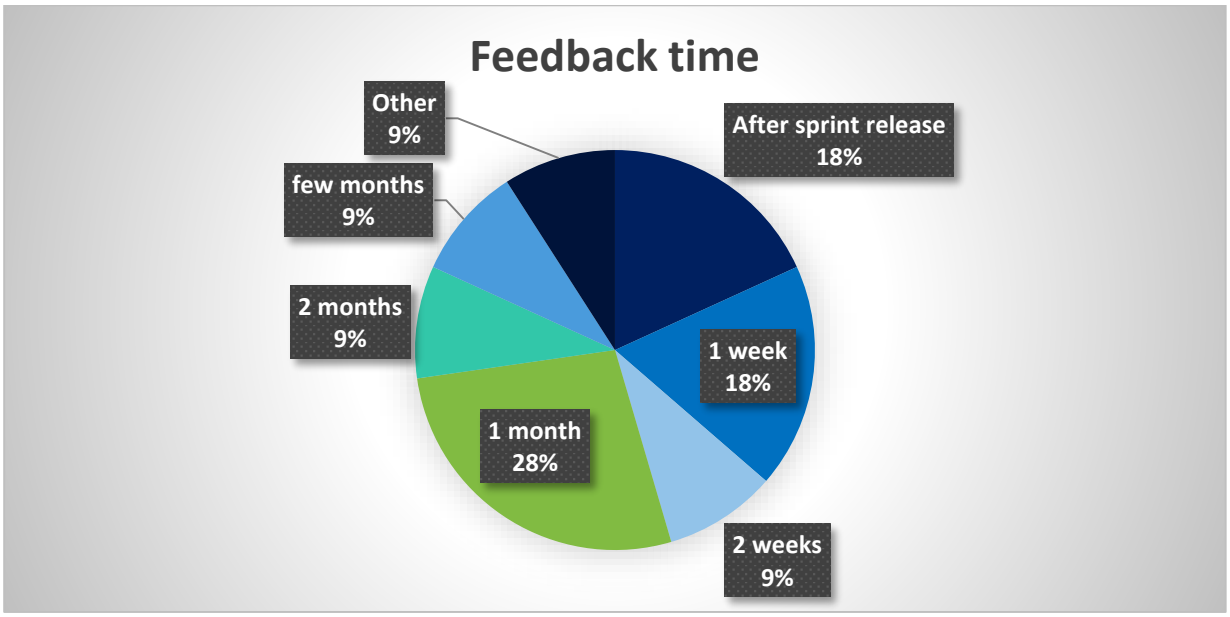

Figure 31: Feedback time

Finally, we analyzed the feedback provided by the customers and its impact on the development or testing process according to the studied organizations. In most of the cases, the customer feedback has impacted various areas related to testing. Six out of 11 organizations have indicated that the feedback provided by the customer usually impacts the test aspects. These aspects include increasing the test coverage, more test automation and more time allocated for testing. The distribution of impacted areas is presented below in figure 32.

Subsequently, two organizations have indicated that the feedback impacts the development process. Specifically the implementation of code reviews and refactoring are mentioned. Furthermore, change of scope and collaboration with the stakeholders have been identified as areas that are impacted by customer's feedback.

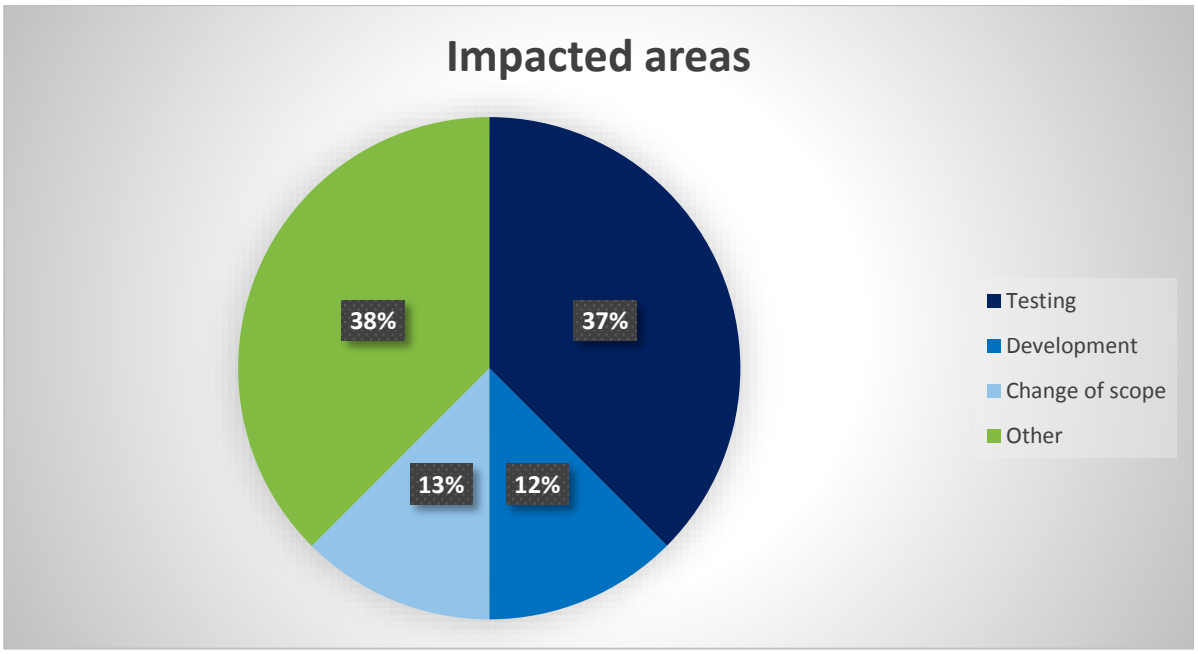

Figure 32: Feedback impacted different areas

4.6.3 Measures derived from poor quality

In order to identify the impact of project success on quality metrics, we will investigate measures taken by organizations in case of project failure. Organizations can try to minimize the chance of failure by taking specific measures. In this section we want to discover whether the taken measures have any relation to implementing quality metrics. This could be introducing more quality metrics, changing or excluding some.

The investigation revealed that some participants could not associate any failed projects with their organizations. As a result, due to lack of failed projects in some organizations, we don't have complete data based on measures taken.

Studied organizations have reported various types of measures taken by them in order to prevent the project failures. Based on the data analysis, areas reflecting testing have been mentioned the most. Increasing the testing time and coverage are measures that many organizations have taken. Subsequently, the team size is a measure that is considered imperative. Adding extra resources to the team and increasing the team size are measures taken by studied organizations. A summary of topics related to these measures is listed below.

- More testing (unit test)
- Team size
- Reduce environment dependency
- Fix quickly
- Reduce complexity
- Refactoring
- Better product design
- Create stable teams
- Better requirement analysis

The results related to measures presented above relate to different areas of increasing the quality. However, we found two measures that are related to two quality metrics. One organization has indicated to fix the issues reported quickly. This measure is related to the quality metric fix response

time. The second identified quality metric is related to unit testing. As mentioned already, many organizations have indicated to increase different aspects of testing including unit testing.

4.6.4 Outcome

In this section we focus to answer the RQ4: *"What is the impact of successful or failed projects on quality metrics?*". In the previous sections we analyzed the project success rate within the organization and focused on the successes. Then we identified the types of feedback provided by the customer and finally we investigated the measures taken by organizations when projects failed.

The study clarified that generally participants find it hard to measure project success. It has become evident that the success rate is in general above 70%, the underlying motive for this success rate is mainly related to different aspects of agile development. These aspects include implementing short iterations, small teams, good collaboration between business and IT and many more. In general, this success rate has not triggered organizations to change or implement new quality metrics.

The feedback provided by the customers is in the most cases related to customer satisfaction and critical issues found that needs to be fixed immediately. The timeframe that the feedback is reported varies from immediately when the sprint ends until a few months after the sprint. We acknowledge that fix response time is a very important quality metric for customer feedback. The customer feedback impacts in most of the cases the testing process. These include increasing the test time, test coverage and implementing automated testing.

We analyzed what measures organizations take when projects fail. The results show that there are a variety of measures that organizations take, in most of the cases the testing area is affected.

Finally to conclude, based on the three dimensions, generally, failed projects are definitely impacting the use of quality metrics. Typically, quality metrics are improved or introduced to prevent future failures on projects. Organizations are implementing more testing, including automated tests to reduce project failure. There are no reports regarding stopping the use of certain quality metrics. We have not perceived any evidence that successful projects impacts quality metrics. This study identified quality metrics that can impact successful or failed projects. These quality metrics are: fix response time, unit test coverage, number of automated test coverage.

4.7 Maturity levels and quality metrics

We have researched when and how often organizations introduce new quality metrics. In addition, we will analyze the results in sections 3.3 and 3.4 to formulate the answer. The outcome will aid to answer the following research question: "*At what maturity levels are which software quality metrics implemented?".*

In the table below we present the results based on when and how often new quality metrics are introduced in organizations.

Organization	When new quality metrics introduced	How often
A	When management needs it, now code coverage	Rarely
B	No changes	Rarely
C	No changes	No changes
D	Rather improving, it depends on the period, right now focus is satisfaction (unit testing)	Rarely
E	No changes	No changes
F	No changes	No changes
G	No changes	No changes
H	No changes	No changes
I	Rarely, last year TICS	Rarely
J	Triggered from retrospectives and ING	Rarely
K	No changes	No changes

Table 32: Implementing of new quality metrics in organizations

According to the results, introducing new quality metrics seems not be a highly exercised activity in organizations. From the 11 researched organizations, only three have been introducing new quality metrics with a limited focus, these are organizations A, I and J. Organization D is merely improving the existing quality metrics instead of introducing new ones. In organization A, the need for new quality metric is derived from the management. In this case, code coverage was introduced. In general, organization A does not introduce quality metrics very often. In organization D, rather than introducing, organization is more or less improving existing quality metrics. However, this is very much related to the period. At this point, the focus is on improving the customer satisfaction and quality, and therefore the implementation of unit testing is improved. Organization I has been introducing TICS last year. TICS is a software solution developed to improve quality by providing insight regarding the source code and other attributes related to code optimizations. Except the TICS, the introduction of new quality metrics or tools happens rarely in organization I. Organization J has indicated that introducing new quality metrics occurs rarely, but if it occurs, it comes from the retrospectives or imposed by the management.

It's notable to mention that the three organizations that do introduce new quality metrics, are belonging to the high maturity level organizations according to our study. These results can be found in section 3.3.1.

Except the aforementioned four organizations that are involved in introducing new quality metrics in a limited way, the other seven organizations do not introduce any new quality metrics. As it's evident from the table 34, we have labeled them with "no changes". The implementation of new quality metrics or modifying the existing ones, is very much neglected. These organizations have indicated that they stick to existing quality metrics without modifying them.

The results of table 33 could not aid to answer our research question in full. Therefore, we will analyze the results in section 3.3 where we have evaluated the maturity levels of the teams, and in section 3.4 where we identified quality metrics used in organization.

4.7.1 Outcome

To conclude and answer the research question, in table 35 we present the maturity levels and the corresponding quality metrics. The levels are categorized in low, medium and high maturity. The low maturity level consists of quality metrics derived from organizations C and F. The medium maturity consists of quality metrics in organization B, D, E, G, H and K. Finally, quality metrics in high mature organizations are collected from organizations A, I and J.

As it's evident from table 35, organizations with low maturity are collecting fewer quality metrics. During the data collection, the study offered a choice from 13 quality metrics, in addition, organizations could add extra quality metrics. From the standard provided quality metrics in our study, low mature organizations are missing seven quality metrics, these fields are presented in pink color. In the medium and high maturity level, we acknowledge that the quality metrics collected, are almost similar. As a result, we can conclude that, there is a relation between the use of specific quality metrics and maturity levels. This relation is only based between low and high maturity organizations. As presented, specific quality metrics are only applied in medium and high maturity levels and are missing in low maturity level.

Low maturity	**Medium maturity**	**High maturity**
Organizations: C and F	Organizations: B, D, E, G, H, K	Organizations: A, I, J
	Defect count during production	Defect count during production
	Defect count reported by customer	Defect count reported by customer
Fix response time	Fix response time	Fix response time
	Test case count	Test case count
	Lines of code (LOC)	Lines of code (LOC)
Code coverage	Code coverage	Code coverage
Unit test coverage	Unit test coverage	Unit test coverage
Compile failures and build defects	Compile failures and build defects	Compile failures and build defects
Weekly defect arrivals	Weekly defect arrivals	Weekly defect arrivals
Number of failed/succeeded auto test	Number of failed/succeeded auto test	Number of failed/succeeded auto test
	Total number of automated test cases	Total number of automated test cases
Number of open customer problems	Number of open customer problems	Number of open customer problems
	Accuracy of estimates	Accuracy of estimates
	(extra) SIG meter	
		(extra)TICS
		(extra) Code complexity

Table 33: Quality metrics and its relation to maturity levels

5 DISCUSSION

5.1 Reflection on research questions

In the results section, we generated an outcome for every RQ discussed. In the outcome sections, we answered the research questions according to these results. In this section, we will evaluate the research questions and assess the answers and discuss specific findings.

RQ1: *What are the most effective quality metrics that are being used in agile organizations?*
The empirical findings for RQ1 show that in general the use of quality metrics is not very popular. As (Hall & Fenton, 1997) argues, organizations favor a typical set of core metrics, dominated by size and effort metrics, primarily used for resource estimation and productivity, rather than for quality. The metrics found popular in a study conducted by (Hall & Fenton, 1997) are: resource estimation, lines of code, design review data and code complexity. According to our study, we have identified effective quality metrics that are used in agile organizations, these are not similar metrics as found by (Hall & Fenton, 1997). We focused on four dimensions to discover the effective quality metrics. These dimensions are:

- What metrics are used in organizations?
- How is software quality measured?
- What are the most important quality metrics?
- When is a quality metric effective?

The four dimensions have contributed to discover effective quality metrics. As a result, the answer to RQ1 is a set of identified quality metric used in agile context. These metrics are: code coverage, unit test coverage, defects and the number of failed/succeeded automated tests.

RQ2: *What are the most effective agile practices that support maturity?*
To answer RQ2 and find the relevant data, we focused on the following three aspects:

- Which agile practices are applied?
- Which agile practices can determine maturity?
- Which agile practices are necessary?

These three aspects have output a list of practices that are found effective to support agile maturity. We have not found any relevant literature that can support or oppose our findings. The answer to RQ2 is a list of practices shown below. The practices are categorized in three maturity levels. For each level, we describe what the most effective agile practices are in order to support maturity for that level.

High maturity practices

- Continuous deployment
- Story mapping
- Cycle time
- Automated acceptance test
- Customer acceptance test
- Test automation

Medium maturity practices

- Unit testing
- Pair programming
- Velocity

- Collective code ownership

Low maturity practices

- Daily stand-up
- Burn down estimation
- Integrated QA/Dev

RQ3: *How do organizations measure the maturity of their agile software development?*

As the literature suggests, agile maturity can be measured by using models (Ozcan-Top & Demirörs, 2013). In our study, we acknowledged that four organizations are measuring maturity. However, only one organization makes use of models and frameworks in order to assess maturity for their teams. This organization is mainly performing in-house assessment by their own agile coaches. These assessments are conducted on a regular basis and are based on certain models. One of these models is the Tuckman model (Bruce W. Tuckman, 1977). The teams are assessed according to the four stages of the model, and then, it's decided how to improve these teams. Other organizations assess their teams by sending out surveys, creating meeting with the teams and requesting input from the team during the retrospective meetings. As a result, the answer to RQ3 is, organizations don't measure their agility unless they are more mature, and maturity is measured by analyzing teams through meetings, surveys and models – this is only conducted by organizations with higher maturity. Organizations in the range of low and medium agile maturity are not investing any effort in assessing teams on agility. We also acknowledged that organizations introduce agile training and workshops in the initial stage of agile implementation. However, in the later stage, there is no assessment on agile maturity and progress, except for organizations in the higher maturity.

We acknowledge that there is a correlation between team's average experience and the organization's maturity level. Agile experience in low and medium maturity organizations are considered to be balanced between poor and fair. Organizations in the low and medium range of agile maturity are not measuring their maturity. However, the average experience of the team could be an indication for organizations to measure agile maturity. This way, organizations could measure agility by evaluating the average team experience on agile. In our study, we acknowledged that organizations with higher maturity consists of teams with relatively high experience in agile.

RQ4: *What is the impact of successful or failed projects on quality metrics?*

We analyzed the research question from different dimensions. The dimensions are; success rate of the projects, aspects related to customer feedback, and what measures have been taken when projects failed due to poor quality. We acknowledged that in general, the success rate is high in organizations due to implementation of agile methodologies. Our study shows that the impact quality metrics is most perceived when projects fail. In those cases, software quality needs to be improved in order to prevent failed projects. Test automation is an important factor and increases quality (Kile & Inampudi, 2007). In our study we can confirm that, quality is improved by applying more test automation and increasing the automated test coverage.

The answer to RQ4 is that generally, only failed projects are impacting the use of quality metrics, quality metrics are improved or introduced to prevent future failures on projects. In addition, there is more effort invested in testing. We have not perceived any impact of successful projects on quality metrics. Finally, this study identified quality metrics that are impacted by failed projects. These quality metrics are: fix response time, unit test coverage, number of automated test coverage.

RQ5: *At what maturity levels are which software quality metrics implemented?*
We analyzed which quality metrics are implemented at which maturity level. The answer to this RQ depends on the answers given to RQ1-RQ4. The literature did not provide any support on this RQ, in order to answer this RQ, we analyzed data related to RQ1-RQ4. As a result, the answer to RQ5 is that, there is a relation between the use of specific quality metrics and maturity levels. This relation is found between on one side, low, and on the other side, medium and high maturity levels. There is a distinct difference between these two sides in terms of implementing quality metrics. We perceive that there are a number of quality metrics implemented in medium and high maturity organizations that don't exist in low maturity organizations. These quality metrics are listed below:

- Defects during production
- Defects reported by customer
- Test case count
- Lines of code
- Number of automated test cases
- Accuracy of estimates

We perceive that the quality metrics implemented in medium and high maturity level are identical and there are no major differences between medium and high maturity levels. The overview of quality metrics with the corresponding levels is presented in table 35, in section 4.7.1.

Main Research question: *How are the maturity of an agile software development approach and the use of particular software quality metrics related?*

Finally, based on the five aforementioned sub RQ's, we can answer the main RQ. We can conclude that, there is a relation between agile and the use of software quality metrics to some extent, this relation is mostly apparent when looking at low and high maturity levels. Organizations with higher maturity are focused to increase quality, using quality metrics related to testing; especially the number of failed/succeeded automated tests and automated test coverage are the applied metrics. On the other hand, organizations with lower maturity are not focusing on quality metrics, they are mainly busy to improve their existing agile practices. For these organizations, the focus is on the number of defects, which is an indicator for quality, and trying to improve their existing agile processes.

5.2 Agile maturity

In this study we reviewed literature in order to understand the different levels of agile maturity and the specific details of each maturity level. These details are converged into level focus points as described in section 2.3.1. We could identify several similarities between the models, but mostly, the level focus points of each model were different. For example, level three of Scrum Maturity Model (Yin & Figueiredo, 2011) describes customer relationship as level focus point, whereas, Agile Adoption Framework (Qumer & Henderson-Sellers, 2009) describes this in level four. However, these focus points of levels are the foundation of our data collection. We have constructed the questions for maturity score based on these level focus points, in accordance with related research questions. As a result, these level focus points were also the driver for score calculation of maturity as described in section 3.6.1. On the calculation side, some of the question can score more points due the fact that they are described as important factors in the studied maturity models. In our data collection phase we have acknowledged that these questions are indeed considered important in organizations to achieve a higher level of agile maturity. The leading factors that were verified according to our results are: test automation, continuous delivery, continuous improvement and other practices mentioned in section 4.4.2.

For the calculation and categorizing agile maturity of organizations, maturity is divided in three levels as mentioned in section 3.6. The levels consist of low, medium and high maturity. This is a more simplified model to assess maturity and is based on the score, instead of specific practices or activities related to certain levels as the models we have reviewed in this study describe. Nevertheless, this simplified model does make use of the practices and activities, but only to provide a score that eventually will determine maturity. It's imperative to mention that the core objective of this study is not to design a model to determine maturity. As a result, we could have used one of the existing models to determine maturity. However, using such a model would not have any link to our research questions, and in addition, it would have not been as accurate. Because, these models are not providing specific measures to determine maturity, but solely indicating the key points to adhere to. The model used in this study divides maturity in three levels with a minimum score for each level. This approach has been simplifying the measurement and provides an assessment which is closer to the topics of this study, which is agile maturity and software quality metrics.

In general, the existing maturity models (Ozcan-Top & Demirörs, 2013) assess maturity on basis of one or two dimensions. These assessments are conducted by simply analyzing which agile practices are implemented in organizations. In our opinion, determining maturity based on one or two dimensions is not very effective, especially considering software quality metrics. As a result, this study formulated 25 questions representing five dimensions. The variety of dimensions will reveal certain aspects and will provide more value to determine agile maturity more accurately in the context of this study. The first dimension is agile practices; we investigated which agile practices are implemented. Implementing an agile practice doesn't mean that is implemented in the right way. Therefore, we picked some important agile practices and investigated to what extent are these practices implemented. The other dimensions are software quality, testing process, software deliveries and organizational strategy. Especially on the organizational strategy topic, we investigated how organizations cope with continuous improvement, agile training and coaching. These topics are broadly discussed in the literature we reviewed, as described in the section 2.3. These dimensions have contributed to a more accurate calculated agile maturity. We cannot confirm whether adding more dimensions will contribute to a more precise maturity assessment, however, these dimensions have helped us to measure maturity from a broader view, yet useful.

5.2.1 Organizational strategy

From the maturity score results we acknowledge that many organizations scored less or no points at all regarding the organizational strategy dimension. This dimension is related to continuous improvement and learning as also discussed in Agile maturity map (Packlick, 2007), Agile Adoption and Improvement Model (Qumer & Henderson-Sellers, 2009) and Maturity model for software development organizations (Soares & Meira, 2013). This is an interesting dimension that shows the relation between awareness of agility and the willingness to improve.

It's evident from the results that many organizations don't know "how agile" they are, simply because they are not performing any assessments to measure their maturity and uncover improvement areas. They are aware on a high level what the improvement areas are, but the details and specifics of it are lacking. In addition, seven organizations are not providing any continuous trainings or workshops to improve on agile. It's interesting that these organizations have the ambition to grow in agile development, but, first of all, they are not conducting any assessments to see where they stand on agile implementation, and secondly, they don't provide any trainings or workshops continuously. Although, almost all organizations have been providing training during the initial stage of agile implementation. These trainings took a few days to maximum a week, and the objective was to get familiar with agile implementation.

We agree with (State of agile survey, 2014), that having a training program, common tools and an internal agile support group is essential for agile growth. In our study, only four organizations have been providing the teams with training and workshops from initial implementation until now. From the four organizations that provide continuous training, only one organization, which is organization A, is investing extensive effort in training and improving teams continuously. Organization A has scored a relatively high maturity score and can be considered mature. Teams in organization A are assessed regularly and are provided with the necessary help and advice from the agile coaches to improve. A similar study has shown that continuous and hands-on training is more preferable to once-off training (Conboy, Coyle, Wang, & Pikkarainen, 2010). As a result, organizations that are providing continuous training, are not only scoring a higher maturity score, but the team members agree with this strategy and find it more beneficent. It's imperative to mention that not every team needs training, according to one of the agile coaches of organization A: "*at some point, teams don't need any training anymore, because they have been evolving so well that they are not following scrum by the book anymore".*

5.2.2 Agile maturity vs experience in agile

The teams of 11 organizations have been assessed in this study, and we generated a total score. Next to that, we investigated the experience of all organizations with agile methods. This analysis is described in section 4.2.1. The purpose of this analysis is to define the scale of the organizations in terms of experience in agile. We specifically looked for organizations that were in the range of just started implementing agile and organizations that have been implementing agile for 5+ years. As a result, we wanted to discover the relation between agile maturity and the experience in agile. With this analysis we want to discover if there is any correlation between organizations with the most experience in years and high maturity score.

There are two organizations that have eight years of experience with agile, and they have not scored the highest maturity score. In contrary, one of these two organizations has scored low to medium score. Especially in organization K, experience is not an indication of maturity. From our study we acknowledged that organization K is a very turbulent organization in terms of shifting teams. The respondent of organization K has indicated that, the teams are usually not stable, there exists no velocity. Team members are pulled out of one team and added to another team to handle the critical

situations. In addition, team members of organization K consist more than 80% of contractors. The contractors are usually working for short periods. Perhaps we could link this instability in teams and working with non-permanent team members with the lower maturity score. This instability is not only affecting agile maturity, but also the productivity and project success. We agree with (Drurya, Conboy, & Power, 2012), that this behavior causes implications, such as not completing the planned work, scope decisions are impacted and when team members are pulled out; there is no additional time left for someone else to cover their work. On the other hand, organization F has the lowest experience with agile and has scored the lowest maturity score. In this case there is a correlation between experience and agile maturity.

5.3 Quality metrics, necessary and neglected

In section 4.3 we presented the collected quality metrics. For the data collection, this study selected 13 metrics mentioned in the reviewed literature (section 2.5), that are relevant to quality and agile maturity. The data analysis has shown that, next to the provided metrics, organization are using tools to manage quality, these tool provide more metrics and insight in quality. There are variety of used to collect metrics, such as Sonar, SIG meter, Fortify and TICS. Based on 22 interviews and surveys conducted, the general impression is that, there is less attention paid to quality metrics and collecting it. According to a respondent from organization A; *"We collect data but it's not our driving force*". A respondent from organization I has indicated; *"Metrics are just number, they provide no value".* And finally, a respondent from organization H has said; "*We use Sonar, but we do nothing with it*".

However, the use of quality metrics is not fully neglected; organizations do gather and pay attention to metrics that are important to their environment. We analyzed the most important quality metrics according to the respondents. The results show that, 9 out of 11 organizations have indicated that defects are the most important followed by code coverage and unit testing. However, the question remains, are these agile metrics? Agile metrics are measures related to agile practices. The most common agile metrics is velocity. Next to velocity, other agile metrics are pulse (measuring continuous integration, product size (amount of completed work), burn (remaining work vs human resources) and faults (number of faults per iteration), (Anderson, 2005; Dubinsky, Talby, Hazzan, & Keren, 2005). According to studied organization is defects considered as one of the most important quality metrics. According to (Dubinsky, Talby, Hazzan, & Keren, 2005), defects can be considered as an agile metric, we agree with that, if it's measured per iteration. Generally, the difference between traditional and agile metrics is that; agile metrics are more focused on measuring progress (Misra & Omorodion, 2011). To conclude, we can acknowledge that there is a broad interrelation between agile and traditional metrics, but mostly, the traditional metrics are used in agile context.

5.3.1 Measuring software quality

We analyzed how organizations measure software quality. As (Hendriks, Vonderen, & Veenendaal, 2000) argues, evaluation of software quality is difficult, in our study we perceived the same. Many organizations don't' know how they measure software quality. In general, organizations have indicated that the number of defects is the leading indicator for measuring software quality. Most of the organizations use the combination of automatically and manually generated data. However, there are no fixed criteria to measure software quality. It remains interesting to perceive that many experts and team members don't exactly know how software quality is measured.

5.3.2 Introducing new quality metrics

In high maturity organization the use of metrics is expected to play a key role (Jalote, 2002). We don't entirely agree with that, according to our study, metrics play a role, but not as important as

described by (Jalote, 2002). In our study we perceived that introducing new quality metrics in organizations does not happen regularly, except for the ones in higher maturity. The results regarding this are described in section 4.7. We acknowledged that from 11 organizations, only three organizations are introducing new quality metrics. These organizations have scored a relatively high maturity score, and we could perceive that there is a correlation between maturity and the use of quality metrics. These results raise an interesting question; *"Does implementing quality metrics helps to become more mature, or, because organizations are mature, they implement quality metrics?*" Based on the results, we cannot perceive any correlation between the amount of quality metrics and maturity. According to (Krebs, Kroll, & Richard, 2008), measuring too many metrics does not contribute to project success. It doesn't mean that necessarily implementing more quality metrics will contribute to success and higher maturity.

Organizations that have medium maturity can be using same amount of metrics or even more. As a result, a model with three maturity levels cannot highlight in detail specific quality metrics related to certain maturity levels. For this aspect, a maturity model with more levels could aid to identify which quality metrics belong to which maturity level However, based on the results presented in section 4.7, we perceive that there is a small correlation between the use of specific quality metrics and maturity levels and necessarily the amount of metrics. Especially, between low and high maturity organizations regarding implementing specific quality metrics. As a result, organizations are implementing quality metrics, because of the fact that they are mature, and not implementing more quality metrics to grow in maturity. In our study we have not received any indication from the respondents and results that quality metrics can improve maturity, except for the fact that certain quality metrics are used more in medium and high maturity organizations.

5.4 Challenging questions

During the data collection phase we conducted interviews. The interviews contained of challenging questions that were hard to answer. One of the most challenging question was "*when is a quality metric effective?*". We perceived that many respondents found it hard to respond to this question. At first, they expressed that the question was unclear, as a result, we elaborated to make the question clearer. Although they understood what exactly the question is, not all of the respondents could provide us with an answer. It raises the question; why is this so hard to answer? Unfortunately, we could not find the underlying factors related to the difficulty of this question. But it's worth mentioning that some of the respondents found it hard to respond to this.

Another challenging question is related to measuring project success. We asked the respondents; *"what is the success rate of projects?".* Most of the respondents had a general idea to what extent the success rate is, but they found it hard to express it in figures. We agree with (Highsmith, 2002), in each organization, projects are managed differently and success is measured differently. In general, project success is achieved if software quality is high, the customer is satisfied, and the products have been released on time. Project success rates provided by respondents are based on rough estimation and it's hard to proof their validity. Some of the respondents have indicated that the project success is 100% due to the fact that they have never been involved in failed projects and that their current project is running for years. As a result, we can only conclude that the project success is applicable to the corresponding team and not the entire organization.

5.5 Agile practices

5.5.1 Difficult practices

In section 4.5.5 we analyzed the improvement areas for agile teams. Seven out of 11 organizations have indicated that planning, followed by retrospective and defining users stories are considered to be difficult and needs improvement. A case study performed on agile describes that, planning can be difficult and sometimes frustrating for the teams (Layman, Williams, & Cunningham, 2006). We agree with (Layman, Williams, & Cunningham, 2006) regarding the difficulty of planning. Our results indicate that planning is considered to be time consuming and difficult to conduct in some cases. One of the factors is related to planning too tight, and therefore risking a failed sprint. Another factor indicated by respondents is that, some of the issues such as defects cannot be planned, and therefore it impacts the planning session with implications. The planning session is impacted because of incomplete user stories. User stories are not defined properly, and during the planning session, the team attempts to refine the user stories and make them fit in the sprint. However, according to the results of this study, defining good user stories, creating small and achievable user stories seems to be a challenge. Finally, retrospective is considered to be an agile practice that four organizations struggle with it. A scrum master from the studied organizations said; "*it usually takes time before everybody opens up".* Another aspect related to retrospective mentioned by respondents is that action points are neglected. At the end of retrospective, action points are created, but not further processed by the team members. One of the respondents said; "*action points are created, but they hang in the air"*. Based on these facts, we could conclude that planning, defining user stories and retrospective can be considered as one of the most difficult agile practices.

5.6 Team experience

Experience in agile team is crucial and a key factor for success. Based on the results described in section 4.5.3, it is evident that team maturity plays an important role. Some of the teams have less experience in agile, whereas other teams are excelling in agile implementation. This variation can be explained due to individual experience with agile development. We acknowledged two correlations. First, organizations with high maturity level have teams that have on average, more than two year experience with agile. Second correlation shows that, low maturity organizations have teams with poor experience with agile. Poor experience is indicated as less than one year experience with agile. As a result, becoming more mature in agile depends on the experience the individuals have in the team. The more experience the team has, the easier it becomes to implement agile methodologies and achieve organizational goals.

On the other hand, the teams will struggle if there is a team with poor experience and just started implementing agile methodologies, like we acknowledged in organization F. Organization F has not only a team with low experience, but also just started three months ago implementing scrum. As a result, it impacts the growth of the team in agile and implementing agile practices in the right way. Unfortunately we couldn't find any relevant literature that agrees or opposes our results.

5.7 Roles in teams

We agree with (Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Nerur & Balijepally, 2007) that agile methodologies encourage interchangeability of roles. We acknowledge that nowadays it's really moving forward and being applied. However, in practice, this does not happen for all organizations, especially in the range of low to medium maturity. We perceived that mostly organizations with more experience in agile and higher maturity are really applying diverse roles for a single team member. In most of the cases the scrum master role was combined with other activities and roles

such as developer or tester. Another interesting practice is the introduction of Devops. There is no standard definition for Devops. Devops brings the QA engineer job description closer to that of developer (Roche, 2013), it's more about the culture and the manner of working. According to (Swartout, 2014), continuous delivery and Devops are the next big thing. We studied 11 organizations, and from those organizations, only one is implementing Devops. The only organization that has Devops, is organization J. In addition, this organization has scored the highest maturity scored in our study. When we asked a team member of this organization what his role is, he replied: *"most of us are Devops".* Every team member is responsible for development, operations and testing. According to the results of our study, we would carefully agree with (Swartout, 2014) to some extent that Devops could be the next big thing.

5.8 Recommendations

Our study has shown that many organizations don't measure how agile they are. Measuring agility and improving agile related processes can increase quality. Agile coaches or scrum masters are the responsible individuals for good implementation of agile methodologies and guarding agile principles in organizations. Agile coaches and scrum masters should conduct more assessments in organizations to identify weak and strong teams in implementing agile. They can take the good practices from the strong teams, and apply these on the weak teams in order to improve the performance. In addition, theoretical models can provide a substantial foundation to measure teams and helping them to overcome their weaknesses. The question remains, how can agile coaches and scrum masters conduct assessments? The empirical evidence from our study shows that, the best moment is during the retrospectives, agile coaches should make use of the opportunity, and request the team to fill out a survey and collect the necessary information to measure agility.

According to our study, the process of measuring software quality seems to be vague. Often organizations rely on metrics such as defects and customer complaints. However, there is little effort invested in collecting structured feedback from the customer. The customer is this context the end-user and not the product owner. This process is very informal and often neglected. Introducing consistent processes to collect feedback from the customer, and communicate this to the team, will contribute to a better product and decreases the chance of failure.

In some of the cases we have perceived that the organizational strategy is not well communicated to the teams. As a result, the teams are unaware of the strategy that the organization is implementing. For example, one of the respondents replied: "*I don't know that what the organizational strategy is, I think it's something with continuous delivery*". Communicating the right messages to teams can help organizations to involve the teams more, and enable commitment from the teams, in order to achieve organizational strategies.

Empowering teams is a fundamental aspect of agile. However, in some cases this does not take place. Particularly in one case, a respondent complained that the scrum master had "too much power". The scrum master was assigning tasks and responsibilities to the team during the stand-up. As a result, the team was unhappy with assigned tasks and could not perform well, because the tasks were assigned to the wrong individuals. Agile teams should be empowered, they are the main responsible individuals for delivering on time with high quality. The teams should not be imposed to use certain tools if they don't want it. Teams will perform better if they choose how to work and with what tools.

Agile coaches and scrum masters should enable variations in certain agile practices. Teams get often bored when they always perform retrospectives in the same way. Conducting retrospectives in different ways can help teams to reflect on their performance from different dimensions, and provide a more in-depth analysis.

6 CONCLUSIONS

Organizations that have been implementing agile for a couple of years, can be considered becoming mature. However, it remains unclear how agile maturity affects software quality. As a result, this study focused on answering the following main research question: "*How are the maturity of an agile software development approach and the use of particular software quality metrics related?".* To discover the empirical findings related to our study, we conducted in total 22 interviews and 22 surveys, across 11 organizations in the Netherlands, United Kingdom and Israel. We spoke to experts and representatives of agile teams. In order to construct a valid data structure to aid answering the main research question, we constructed five research sub-questions.

The findings of this study are presented in the results section, concerning the five research questions. In the results section, the corresponding chapters end with an outcome, where we answer the research question relevant to that section purely based on data. Later in the discussion, section 5.1, we assessed the research questions and concluded the given answers. An overview is provided in the table below.

Research question		In this thesis
Main RQ	*How are the maturity of an agile software development approach and the use of particular software quality metrics related?*	Section 5.1
RQ1	*What are the most effective quality metrics (QM) that are being used in agile organizations*	Section 4.3
RQ2	*What are the most effective agile practices that support maturity*	Section 4.4
RQ3	*How do organizations measure the maturity of their agile software development?*	Section 4.5
RQ4	*What is the impact of successful or failed projects on quality metrics?*	Section 4.6
RQ5	*At what maturity levels are which software quality metrics implemented?*	Section 4.7

Table 34: Overview of research questions

In this section we provide a short summary of aforementioned answered research questions.

With RQ1 we tried to discover the most effective quality metrics in agile organizations. We have identified these metrics. As a result, the answer to RQ1 is a set of identified quality metric used in agile context. These metrics are: code coverage, unit test coverage, defects and the number of failed/succeeded automated tests.

RQ2 was stated to discover the most effective agile practice to support agile maturity. Our study has identified agile practices that are found effective in order to support agile maturity. We categorized these practices in three levels. The answer to RQ2 consists of three levels with corresponding practices. Practices to support low maturity are: Daily stand-up, Burn down estimation, Integrated QA/Dev. Practices for medium maturity are: Unit testing, Pair programming, Velocity, Collective code ownership. Finally, practices for high maturity are: Continuous deployment, Story mapping, Cycle time, Automated acceptance test, Customer acceptance test, Test automation.

RQ3 is related to how organizations measure agility. The answer to RQ3 is: organizations don't measure their agility unless they are more mature, and maturity is measured by analyzing teams through meetings, surveys and models – this is only conducted by organizations with higher maturity. Organizations that are considered low or medium mature, are not investing any effort to measure agile maturity

RQ4 is concerned with how project success affects software quality. We analyzed this RQ from different dimensions. The answer to RQ4 is: generally, only failed projects are impacting the use of quality metrics, quality metrics are improved or introduced to prevent future failures on projects. In addition, there is more effort invested in testing. Finally, this study identified quality metrics that are impacted by failed projects. These quality metrics are: fix response time, unit test coverage, number of automated test coverage.

RQ5 investigates the relation between specific quality metrics and agile maturity. The answer to RQ5 is that, organizations with medium and high maturity are implementing quality metrics that don't exists in low maturity organizations. These metrics are: Defects during production, Defects reported by customer, Test case count, Lines of code, Number of automated test cases, Accuracy of estimates. In addition, quality metrics implemented in medium and high maturity level are identical, and there are no major differences between medium and high maturity levels.

Finally, the answer to main RQ is: there is a relation between agile and the use of software quality metrics to some extent, this relation is mostly apparent when looking at low and high maturity levels. Organizations with higher maturity are focused to increase quality, using quality metrics related to testing; especially the number of failed/succeeded automated tests and automated test coverage are the applied metrics. Organizations with lower maturity are not focusing on quality metrics, they are mainly busy to improve their existing agile practices.

In general, the use of quality metrics is not a very common practice, especially in organizations with lower maturity. Quality metrics such as defects, code coverage, unit test coverage and automated tests are found to be the most popular and meaningful by organizations. Other than that, tools are providing a lot of data regarding the quality, but that seems to be neglected. However, the use of metrics is more conducted in organizations with a higher maturity. These organizations are focusing on quality metrics such as unit test coverage, automated test coverage and delivering high quality software. Organizations that become more mature, produce higher quality software due to effective implementation of agile. Organizations that have become mature, don't implement scrum by the book anymore. Autonomy and empowering teams allows them to develop their own structure of agile implementation. Teams at this stage have become mature and the focus on quality is increased. As a result, agile maturity leads to higher software quality, indicated by effective implementation of agile practices related to software quality.

6.1 Future work

Further empirical work with larger and broader samples will help us to map quality metrics better to certain maturity levels. We conducted our study in 11 organizations with two representatives of each team. We recommend to conduct the future research in more organizations to map broader set of maturity levels, and interview all team members. Investigating team's experience in agile and its relation to agile maturity would be an interesting topic to research. Especially, in organizations where Scaled Agile Framework (SAFE) is implemented, the growth of the teams and their experience with agile is interesting to research. Furthermore, future research based on agile maturity quality metrics should include aspects such as definition of done and technical debt.

7 STRENGHTS AND WEAKNESSES

A strength of this study is that it's the first study to investigate two crucial aspects of agile, which is maturity and quality metrics, including all dimensions that can affect software quality. Another strength is that, we conducted our research in three countries with different cultures. This a different dimension and provides results that can be used as a foundation for a general understanding. This only not provides results from a bigger picture, but also takes into account the cultural aspect in different countries. Another strength is that we selected organizations based on experience in agile. We studied organizations with almost zero experience, to organizations that have eight years of experience in agile development. Using this, we defined a scale from low to high in order to construct a base for maturity levels.

This study has several limitations as well. One of the limitation is related to the qualitative design of our multiple case study. Due to this, the perception of the participants can be different, and they can have a biased view on their work process and therefore could be hard to be validated. Furthermore, some of the questions for the survey and interview could not be answered due to lack of expert knowledge. Adding more questions to the survey and interview will provide a more detailed and constructive maturity score. However, this could be hard due to time limitations. Although we had 22 participants in 11 organizations, our sample might be hard to be reproduced.

7.1 Validity considerations

To perform a valid study, we have encouraged the participants to provide honest and realistic answers. In addition, we stressed that the data will be treated anonymously to fully encourage the level of honesty from the participants. To provide transparency, we have included the full data analysis in detail. Finally, data analysis and testing included the four criteria: construct validity, internal validity, external validity and reliability (Yin R. K., 1994).

To address construct validity, the pre-defined set of interview and survey questions with their relation to specific research questions, allowed us defining a valid initial starting point. The semi-structured interviews aided to analyze aspects from different angles and dimensions. Furthermore, we kept the data analysis phase consistent to the methodology described.

As for the internal validity, in this study, we used different ways of visualizing the data to discover patterns and matches. In addition, the use of tables allowed presenting the data in a clear manner.

To address external validity, we repeatedly mentioned that the results are not reflective for the entire organization, but solely for the team researched. We have collected data in 11 organizations in the same manner and methods. Within the agile team, we interviewed different roles. We have clearly stated the research strategy, which is based on certain criteria.

The methodology applied in this study, clearly states what interview questions are used, how the survey is conducted, and how the data analysis is performed. For the data collection phase, we followed the exact same procedure for all organizations.

8 BIBLIOGRAPHY

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods. *VTT PUBLICATIONS* .

Agile manifesto. (2001). Agile manifesto.

Alegria, J. A., & Bastarrica, M. C. (2006). Implementing CMMI using a Combination of Agile Methods.

Ambler, S. (2005). Quality in an agile world.

Ambu, W., Concas, G., Marchesi, M., & Pinna, S. (2006). Studying the Evolution of Quality Metrics in an Agile/Distributed Project.

Anderson, D. J. (2005). Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI® Process Improvement at Microsoft Corporation .

Begel, A., & Nagappan, N. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context:An Exploratory Study. *First International symposium on empirical software engineering and measurement*, 255-264.

Botella, P., Burgués, X., Carvallo, J., Franch, X., Grau, G., Marco, J., & Quer, C. (2004). ISO/IEC 9126 in practice: what do we need to know?

Bruce W. Tuckman, M. A. (1977). Stage of small group development revisited. 419.

Cardozo, E. S., & Neto, J. B. (2009). SCRUM and Productivity in Software Projects: A Systematic Literature Review.

Cheng, T.-H., Jansen, S., & Remmers, M. (2009). Controlling and Monitoring Agile Software Development in Three Dutch Product Software Companies.

Conboy, K., Coyle, S., Wang, X., & Pikkarainen, M. (2010). People Over Process: Key People Challenges in Agile Development.

CrunchBase. (n.d.). *CrunchBase*. Retrieved from https://www.crunchbase.com/organization/causata

Davis, N. (2013). Driving Quality Improvement and Reducing Technical Debt with the Definition of Done.

Drurya, M., Conboy, K., & Power, K. (2012). Obstacles to decision making in Agile software development teams. *Elsevier, The Journal of Systems and Software*.

Dubinsky, Y., Talby, D., Hazzan, O., & Keren, A. (2005). Agile Metrics at the Israeli Air Force.

Dyba, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review.

Fowler, M. (1999). Refactoring: Improving the design of existing code.

Grinyer, A. (2007). Investigating Adoption of Agile Software Development. *Agile process in software engineering and extreme programming*, 163-164.

Hall, T., & Fenton, N. (1997). *Implementing effective software metrics programs.*

Hendriks, R., Vonderen, R. v., & Veenendaal, E. v. (2000). Measuring software product quality during testing.

Highsmith, J. (2002). What Is Agile Software Development. *CROSSTALK The Journal of Defense Software Engineering*.

IEEE. (1998). IEEE Std 1074 -1997 - Standard for Software Life Cycle Processes.

Imreh, R., & Raisinghani, M. S. (2011). Impact of Agile Software Development on Quality within Information Technology Organizations . *Journal of Emerging Trends in Computing and Information Sciences*.

ISO 8402. (1986). ISO 8402 Quality Vocabulary - in International Organization for Standardization.

ISO/IEC 9126. (2001). ISO/IEC 9126-1 Software engineering- Product quality- Part 1-4.

ISO/IEC 9126-2 . (2001). ISO/IEC 9126-2 Software engineering -Product quality- Part 2: External metrics.

ISO/IEC 9126-3. (2001). ISO/IEC 9126-3 Software engineering -Product quality- Part3: Internal metrics.

ISO/IEC 9126-4. (2001). ISO/IEC 9126-4 Software engineering -Product quality- Part 4: Quality In Use metrics.

Jalote, P. (2002). Use of Metrics in High Maturity Organizations .

Janus, A., Schmietendorf, A., Dumke, R., & Jäger, J. (2012). The 3C Approach for Agile Quality Assurance.

Jeon, S., Han, M., Lee, E., & Lee, K. (2011). Quality Attribute driven Agile Development.

Jinzenji, K., Williams, L., Hoshino, T., & Takahashi, K. (2013). Empirical study of Software Quality Evaluation in Agile Methodology Using Traditional Metrics.

Jyothi, V. E., Srikanth, K., & Rao, K. N. (2012). EFFECTIVE IMPLEMENTATION OF AGILE PRACTICES – OBJECT ORIENTED METRICS TOOL TO IMPROVE SOFTWARE QUALITY.

Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering.* Boston: Addison-Wesley Longman Publishing Co.

Kassim, N., & Zain, M. (2004). Assessing the Measurement of Organizational Agility. Journal of American Academy of Business.

Kile, J. F., & Inampudi, M. R. (2007). Agile Software Development Quality Assurance: Agile Project Management,Quality Metrics, and Methodologies.

Kohlegger, M., Maier, R., & Thalmann, S. (2009). Understanding maturity models resultsof a structured content analysis.

Kumar, G., & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*.

Kumar, G., & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, Volume 2.

Kunz, M., Dumke, R. R., & Schmietendorf, A. (2008). How to Measure Agile Software Development.

Layman, L., Williams, L., & Cunningham, L. (2006). Motivations and Measurements in an Agile Case Study.

M.Sarshar, R.Haigh, M.Finnemore, G.Aouad, Barrett, P., Baldry, D., & Sexton, M. (2000). SPICE: A Business Process Diagnostics Tool for Construction.

Misra, S., & Omorodion, M. (2011). Survey on Agile Metrics and Their Inter-Relationship with Other Traditional Development Metrics.

Moe, N. B., Dingsøyr, T., & Røyrvik, E. A. (2009). Putting Agile Teamwork to the Test – An Preliminary Instrument for Empirically Assessing and Improving Agile Software Development.

Moser, R., Abrahamsson, P., Pedrycz, W., Sillitti, A., & Succi, G. (2008). A case study on the impact of refactoring on quality and productivity in an agile team.

Nerur, S., & Balijepally, V. (2007). Theoretical Reflections on AGILE DEVELOPMENT METHODOLOGIES. *COMMUNICATIONS OF THE ACM March 2007/Vol. 50, No. 3*.

NICE.com. (n.d.). *NICE.com*. Retrieved from http://www.nice.com/company-overview

Ozcan-Top, O., & Demirörs, O. (2013). Assessment of Agile Maturity Models:A Multiple Case Study.

Packlick, J. (2007). The Agile Maturity Map A Goal Oriented Approach to Agile Improvement.

Patel, C., & Ramachandran, M. (2009). Agile Maturity Model (AMM): A Software Process Improvement framework for Agile Software Development Practices.

Paulk, M. C. (1999). Analyzing the conceptual relationship between ISO/IEC 15504 (software process assessment) and the capability maturity model for software.

Paulk, M. C., Konrad, M. D., & Garcia, S. M. (1995). CMM Versus SPICE Architectures .

Qumer, A., & Henderson-Sellers, B. (2009). A framework to support the evaluation, adoption and improvement of agile methods in practice.

Roche, J. (2013). Adopting Devops practices in Quality Assurance.

Sato, D., Goldman, A., & Kon, F. (2007). Tracking the Evolution of Object-Oriented Quality Metrics on Agile Projects.

Schwaber, K., & Sutherland, J. (2013). Scrum guide.

Sfetsos, P., & Stamelos, I. (2010). Empirical Studies on Quality in Agile Practices: A Systematic Literature Review.

Sidky, A., & Arthur, J. (2007). A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework.

Sirshar, M., & Arif, D. F. (2012). Evaluation of Quality Assurance Factors in Agile Methodologies. *International Journal of Advanced Computer Science, Vol. 2*, 73-78.

Soares, F. S., & Meira, S. R. (2013). An Agile Maturity Model for Software Development Organizations.

State of agile survey. (2014). State of agile survey.

Stettina, C. J., & Heijstek, W. (2011). Five Agile Factors: Helping Self-management to Self-reflect.

Stettina, C. J., & Hörz, J. (2015). Agile portfolio management: An empirical perspective on the practice in use. *International Journal of Management*, 140-152.

Swartout, P. (2014). *Continuous Delivery and DevOps – A Quickstart Guide.* PACKT publishing.

TechCrunch.com. (n.d.). Retrieved from http://techcrunch.com/tag/fizzback/

University of Connecticut. (n.d.). *Likert Scale*. Retrieved from http://www.gifted.uconn.edu/siegle/research/instrument%20reliability%20and%20validity/likert.html

Yin, A., & Figueiredo, S. (2011). Scrum Maturity Model.

Yin, R. K. (1994). *Case study research: design and methods.* Sage Publications.

9 APPENDIX

9.1 A – A sample of survey

Questionnaire

Please take your time and answer honestly.

If you would like to adjust a choice, please draw a circle around the field you would like to mark.

The scale is defined as follows:

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
		Not usually	***Rarely***	***Occasionally***	***Often***	***Usually***		

Date: [_____], Team: [_____]

1. What measures or metrics do you collect? Please also specify all the measures that you take but are not on this list.

- ☐ Defect count during production
- ☐ Defect count reported by customer
- ☐ Fix response time
- ☐ Test case count
- ☐ Lines of code (LOC)
- ☐ Code coverage
- ☐ Unit test coverage
- ☐ Compile failures and build/integration defects
- ☐ Weekly defect arrivals and backlog during testing
- ☐ Number of failed/succeeded automated tests
- ☐ Total number of automated test cases
- ☐ Number of open customer problems
- ☐ Accuracy of estimates
- ☐ Others: _____

2. How do you measure software quality?

- ☐ We don't
- ☐ Automatically generated data using tools
- ☐ Manually generated data
- ☐ We tried collecting metrics but we found them useless
- ☐ We collect it but we do nothing with it
- ☐ We have to, it is part of our process
- ☐ Other: _____

3. How much is the source code covered by unit testing (in percentages)?

_____

4. Does the code often need maintenance?

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

5. Is there any "extra time" given for cleaning up and re-factoring the source code?

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***A great deal***
				Occasionally				

6. Is the test engineer always testing the latest build?

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

7. What agile practices does your team apply?

- ☐ Scrum poker
- ☐ Unit Testing
- ☐ Story Mapping
- ☐ Pair Programming
- ☐ Open Work area
- ☐ Iteration Planning
- ☐ Continuous Deployment
- ☐ Integrated Dev/QA
- ☐ Automated Builds
- ☐ Daily Standup
- ☐ Coding Standards
- ☐ Digital Task board
- ☐ Refactoring
- ☐ Burn down/ Team-Based Estimation
- ☐ Cycle Time
- ☐ Release Planning
- ☐ Automated Acceptance Testing
- ☐ Velocity
- ☐ Agile Games
- ☐ Dedicated Product Owner
- ☐ Continuous Integration
- ☐ Kanban
- ☐ Test-Driven Development
- ☐ Collective Code Ownership
- ☐ Retrospectives
- ☐ Customer acceptance tests
- ☐ Others: _____

8. Do the test engineers make use of automated test scripts?

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

9. What percentage of test scripts is automated?

_____

10. Please specify all the participants in the planning session (e.g. Dev. QA, info analyst etc.).

_____

11. All members of the team actively participated during iteration planning meetings.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

12. All the tasks for the sprint were estimated.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

13. The team missed the sprint deadline.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***

14. Working software was the primary measure for project progress.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

15. The team rather reduced the scope than delayed the deadline.

	1	2	3	4	5	6	7	
Never								***Always***

	☐	☐	☐	☐	☐	☐	☐	
				Occasionally				

16. At the end of iteration, we delivered a potentially shippable product.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

17. How frequently do you release working software? E.g. weekly, monthly.

_____

18. Scrum master was always present during the stand-up.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

19. Stand up meetings were extremely short (max. 15 minutes).

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

20. All relevant technical issues or organizational impediments came up in the stand-up meetings.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

21. In the retrospectives (or shortly afterwards), we systematically assigned all important points for improvement to responsible individuals.

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

22. The team was always sitting together in the same room

	1	2	3	4	5	6	7	
Never	☐	☐	☐	☐	☐	☐	☐	***Always***
				Occasionally				

23. What artefacts are created specifically for people outside of the team?

_____

24. What are currently the focus points of the organization to improve on agile?

_____

25. Do any of the team members have any form of agile certification? E.g. SM, Exin Scrum etc. Please specify which team member has what.

_____

26. What agile practices/techniques would you like to conduct that are currently not in place?

_____

27**.** Is there any freedom from the organization to allow implementation of agile practices?

Never	1	2	3	4	5	6	7	Always

☐ ☐ ☐ ☐ ☐ ☐ ☐

Occasionally

28. How much does the team make use of this "freedom" to implement (new)agile practices? What works and what doesn't? Do you agree? How is your team's way of working agile different from how you think it should be?

_____

_____

29. How long after the sprint ends, you receive feedback from the customer?

_____

30. What percentage of projects is successful?

_____

31. How often do you measure customer satisfaction?

- ☐ Weekly
- ☐ After the sprint release
- ☐ Monthly
- ☐ Other: _____

32. How do you measure customer satisfaction?

- ☐ Reported defects by customer
- ☐ On time software release
- ☐ Feedback request (survey, questionnaire etc.)
- ☐ Other: _____

Thank you for filling out this survey.

9.2 B – Interview questions

A. What is your personal experience with agile?

B. How long has your company been doing agile development? What agile development?

C. What is the length of the projects?

D. What is the duration of the sprints? Why?

E. What tools do you use to manage agile processes and activities? E.g. Jira, confluence, word, excel, post-its, internal wiki etc.

F. What test tools do you use?

G. What do you do well in agile (as a team)?

H. What things don't you do well in agile (as a team)? Or needs improvements?

33. What are the most important quality metrics? Why these?
34. Do you change the quality metrics often? Why?
35. When is a quality metric effective?
36. Is there a continuous delivery pipeline? How does it look like?
37. What agile practices are really necessary?
38. What is the area of improvements for you and your team to use agile methods better?
39. Is the company performing any agile assessment? How? What measures?
40. How familiar are the team members with agile methods? Experience in years?
41. Is there any agile training/workshop provided by the company? How often? For who?
42. What feedback do you receive regarding the quality?
43. What percentage of projects is successful? Why is the success rate of the projects like this?
44. What measures did you take when projects failed due to poor quality?
45. How does the customer feedback change the test or the development process?
46. When do you introduce new quality metrics? Why?
47. How often do you implement new quality metrics?

9.3 C - Improvement areas for teams

Organization	Improvement areas indicated by teams
A	Communication with business, better overview of user stories with related tasks
B	Retrospectives, planning, communication, backlog refinement
C	Planning, clear processes, team responsibility, testing
D	More responsibility, dedication(fix own bugs), retrospective(takes time to open up), SM has too much power(assigning tasks), planning(bugs)
E	Use of test automation, changes during sprint, planning, jumping around products
F	Planning(very long), Jira tasks, backlog refinement, code quality, automated tests(QA and unit test), continuous builds, team commitment, estimation
G	Better user stories, code review(sooner), planning(not perfect), estimates, retrospective(action points)
H	Retrospectives(action points), communication with other teams, user stories(defining)
I	User stories(defining), communicating with other teams, trying be perfect, pair programming
J	Planning, communication with other teams, broader skills, heroes in teams, changes during sprint
K	General administrations

Table 35: Improvement areas in teams

Current quality metrics	Improvements?	Insight?	Measurable?	Proven it works?
Code coverage	x	X	x	-
Unit test coverage	x	x	x	-
# Failed/succeeded auto test	X	x	x	-
# Open customer problems (defects)		x	x	-

Table 36: Current metrics vs characteristics of effectiveness

9.4 D – Overview all quality metrics in literature

Metric	Moser et. al	Sfetsos & Stamelos	Cheng & Jansen	Quality in agile world	Yael Dubinsky et. al	Walter Ambu et. al	Danilo Sato et. al	H. Kan
Coupling between Objects	X					X		
Lack of Cohesion in Methods	X					X	X	
Weighted Methods per Class	X					X	X	
Response For a Class	X					X		
Lines of code	X						X	
Effort	X							
Number of acceptance tests		X						
Total number of defects		X		X				X
Number of defects/KLOC		X						
Number of the defects found before the release		X						
Number of defects reported by the customer		X						X
Code size		X						
Cyclomatic complexity		X					X	
Coupling and cohesion		X						
Total reported defects			X					
Number of critical defects			X					X
Outstanding defects			X					
Fixed/solved defects			X					
Defects coming from previous release			X					
Test failure rate			X					
Hours spent on bug			X					
Test success rate			X					
Product size				X				
Pulse				X				
Burn				X				
Number of Classes					X			
Class Size					X			
Number of Test Cases					X			
Number of Assertions					X			

Response for a Class (RFC)						X		
Depth of Inheritance Tree						X	X	
Number of Children (NOC)						X	X	
Afferent Coupling (AC)							X	
Efferent Coupling (EC)							X	
Mean time to failure								X
Defect density								X
Customer-reported problems								X
Customer satisfaction								X
Phase-based defect removal pattern								X
Defect removal effectiveness								X
Defect density during formal machine testing								X
Defect arrival pattern during formal machine testing								X
Fix backlog								X
Backlog management index								X
Fix response time and fix responsiveness								X
Percent delinquent fixes								X
Defective fixes								X
Compile failures and build/integration defects								X
Weekly defect arrivals and backlog during testing								X
Defect severity			X					X
Defect cause and problem component analysis								X
Reliability								X
Number of CPU hours per system								X
Number of system crashes								X
Models for post-release defect estimation								X

Table 37: Overview of all quality metrics discussed in literature

10 GLOSSARY

SW: software
QM: Quality Metric
RQ: Research Question
ISO: International Organization for Standardization
CMM: Capability Maturity Model
CMMI: Capability Maturity Model Integration
SPICE: Standardized Process Improvement for Construction Enterprises
AMM: Agile maturity model
AMM: Agile maturity map
AAIM: Agile Adoption and Improvement Model
SAMI: Sidky Agile Measurement Index
AAF: Agile adoption framework
SMM: Scrum maturity model
FDD: Feature Driven Development
XP: Extreme Programming
LOC: Lines Of Code
TDD: Test Driven Development
CMS: Content Management System
Dev: Developer
QA: Quality Assurance
Org: Organization
Sig = Sig meter
Tics = TIOBE Coding Standard Framework (TICS)
CC = Code Complexity
B2C = Busines to Consumer

11 ABOUT THE AUTHOR

The author of this thesis document is Mohsen Rezai, student of Master ICT in Business at Leiden University in the Netherlands. This thesis document is the final part of his master's program. Mohsen has earned his Bachelor's degree from the University of Applied Science in Amsterdam. In Amsterdam he studied Technical computing and after his graduation he continued his path by entering the Master ICT in Business at Leiden University. Since 2012, Mohsen has been working as a part time QA engineer in an agile environment. Next to the theoretical foundation, he developed his knowledge in practice working in an international environment, involved in diverse agile projects. Next to researcher's point of view based on the theory, Mohsen has also the practical experience, and can analyze aspects from the practitioner's point of view.