AGILITYMOD: A SOFTWARE AGILITY REFERENCE MODEL FOR AGILITY ASSESSMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZDEN ÖZCAN TOP

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

DECEMBER 2014

AGILITYMOD: A SOFTWARE AGILITY REFERENCE MODEL FOR AGILITY ASSESSMENT

Submitted by **Özden ÖZCAN TOP** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Information Systems, Middle East Technical University** by,

Prof. Dr. Nazife BAYKAL _____
Director, **Informatics Institute**

Prof. Dr. Yasemin YARDIMCI ÇETİN _____
Head of Department, **Information Systems**

Prof. Dr. Onur DEMİRÖRS _____
Supervisor, **Information Systems, METU**

Examining Committee Members:

Prof. Dr. Ali DOĞRU _____
Computer Engineering Dept., METU

Prof. Dr. Onur DEMİRÖRS _____
Information Systems Dept., METU

Assoc. Prof. Dr. Aysu BETİN CAN _____
Information Systems Dept., METU

Asst. Prof. Dr. Kayhan İMRE _____
Computer Engineering Dept., Hacettepe University

Assoc. Prof. Dr. Altan KOÇYİĞİT _____
Information Systems Dept., METU

Date: 05.12.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Özden, Özcan Top

Signature : _____

ABSTRACT

AGILITYMOD: A SOFTWARE AGILITY REFERENCE MODEL FOR AGILITY ASSESSMENT

Özcan Top, Özden

Ph. D., Department of Information Systems

Supervisor: Prof. Dr. Onur Demirörs

December 2014, 148 pages

Agile software development methodologies have gained significant interest in IT community proposing solutions to problems of traditional, plan-driven software development approaches. However, not every organization that tries to adopt agile methods succeeds, that is mostly because practitioners misinterpret the agile values, principles or practices during the adoption and transformation or see a single agile method as a complete solution to all problems. There is a gap in the field to assist software organizations in assessing their agility levels and introducing roadmaps in adopting agile principles/practices. In this thesis study we propose a Software Agility Assessment Reference Model that will be used for assessing organizations' position in agility and indicating the gaps that prevents fully obtaining the benefits of agile software development and providing roadmaps to roadmaps to organizations in adopting agile principles/practices. The model is based on the meta-model structure of ISO/IEC 15504-Process Assessment Standard to create a common basis for performing assessments of agility and present the assessment results using a common rating scale. We performed exploratory case studies and obtained the opinions of the experts to improve the Model. Validation of the proposed model is achieved through one of the qualitative research methods, case studies. We performed a multiple case study including six cases for validation.

Keywords: Agility Assessment Reference Model, Agile Maturity, AgilityMod, ISO/IEC 15504, Agile Software Development

ÖZ

AGILITYMOD: ÇEVİKLİK DEĞERLENDİRME İÇİN BİR YAZILIM ÇEVİKLİK REFERANS MODELİ

Özcan Top, Özden

Doktora, Bilişim Sistemleri

Tez Yöneticisi: Prof. Dr. Onur Demirörs

Aralık 2014, 148 sayfa

Çevik yazılım geliştirme yöntemleri geleneksel, plan odaklı yazılım geliştirme yaklaşımının doğurduğu problemlere getirdiği çözümler nedeniyle BT çevrelerinde önemli bir ilgiyle karşılanmıştır. Fakat her organizasyon çevik yöntemleri uyarlamada tam anlamıyla başarılı olamamaktadır. Bunun nedenlerinden biri çoğu zaman tek bir çevik yöntemin tüm sorunlar için çözüm olarak görülmesi, diğeri ise çevik değer ve prensiplerin yanlış yorumlanmasıdır. Bu alanda organizasyonları çeviklik seviyelerini değerlendirme ve çevik iyileşme yönünde yol haritaları sunma konusunda destekleyecek modellere ihtiyaç vardır. Bu tez çalışmasında organizasyonların çeviklik seviyelerini değerlendirmeye ve boşluk analizi yapmaya olanak sağlayacak bir Çeviklik Değerlendirme Yöntemi geliştirilmiştir. Modelin yapısı ISO/IEC 15504 Süreç Değerlendirme Standardının meta modeli ile uyumlu olarak oluşturulmuştur. Bunun amacı çeviklik değerlendirme için ortak bir temel oluşturulması ve değerlendirme sonuçlarının kabul edilmiş bir notlandırma sistemi üzerinden değerlendirilebilmesidir. Modeli geliştirmek için araştırıcı durum çalışmaları gerçekleştirilmiş, konu ile ilgili uzmanları görüşleri alınmıştır. Önerilen modelin geçerlemesi nitel araştırma yöntemlerinden biri olan durum çalışmaları üzerinden sağlanacaktır. Modelin geçerlenmesi için altı farklı çalışmayı içeren çoklu bir durum çalışması gerçekleştirilmiştir.

Anahtar Kelimeler: Çeviklik Değerlendirme Referans Modeli, Çevik Olgunluk, AgilityMod, ISO/IEC 15504, Çevik Yazılım Geliştirme

dedicated to my beloved husband Can Barış

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Prof. Dr. Onur Demirörs for his great support. He enlightened me with his bright ideas, criticism and insights about the study. I learned a lot from him throughout my PhD not only in academic area but also about life. I know what it means to work hard with passion and enjoy life at the same time by observing him. I also want to thank him for encouraging me joining conferences in our field. This provided me a great vision and allowed me to meet many people.

I would also like to thank my committee members Prof. Dr. Ali Doğru and Assoc. Prof. Dr. Altan Koçyiğit for their ideas and support throughout the thesis study.

I am grateful to Alpay Karagöz, Madhu Parella and Sylvia Trudel for reviewing my work and providing me feedback.

I would like to thank Ece Pekaslan İşel, Doğu Tümerdem, Esin Acar, Engin Ezer, Pınar Efe, Mert Ertuğrul, Ozan Raşit Yürüm, Ali Sağlam and Hakan Kocakulak for their support.

I would like to thank the organizations in which I conducted case studies and their personnel for their contributions to case studies.

I am grateful to my true angels Banu Aysolmaz, Rahime Belen Sağlam and Nurcan Alkış, Meeting you is one of the greatest fortunes in my life. We had wonderful times together that I will never forget. I am greatful to Erdir Ungan and Deniz İren for enriching my life.

I learned that finishing a PhD requires a great deal of persistence, focus, energy, time, creativity, but most important, support. Atilla Soykan has been one of the greatest supporters of my life. I am grateful to him for introducing me a new way of thinking, and teaching me how to live mindfully.

I would like to thank my mother Münevver Özcan, my father Mustafa Özcan, and my beautiful sister Zeynep Özcan for being there for me whenever I needed them and for their constant love. I also want to thank my grandmother Ayşe Güroğlu, my aunt Ülkü Kılıç, my uncles Mustafa Güroğlu, Murat Güroğlu and Adnan Güroğlu for being with me while I am growing up. I am grateful forever.

It is very hard to put my feelings about my husband into words. He was with me for the hardest times. We laughed, worried, grown up and stood up together. Thank you for being with me. Thank you for laughing all my weirdness. Thank you for loving me endlessly and believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AAF	: Agile Adoption Framework
AE	: Agile Elaboration
AgilityMod	: Software Agility Assessment Reference Model
AM	: Agile Modelling
AMD	: Amendment
ASD	: Adaptive Software Development
ASDEs	: Agile Software Development Ecosytems
ASM	: Agile Scaling Model
AP	: Aspect Practice
BM	: Benefield's Model
BP	: Base Practices
CMMI	: Capability Maturity Model Integrated
DSDM	: Dynamic Systems Development Method
ERP	: Enterprise Resource Planning
FA	: Fully Achieved
FDD	: Feature-Driven Development
GAP	: Generic Agility Practice
GP	: Generic Practices
GR	: Generic Resource
GWP	: Generic Work Product
IEC	: the International Electrotechnical Commission
INVEST	: Independent, Value-Added, Small, Estimable, Testable
ISD	: Internet Speed Development
ISO	: the International Organization for Standardization
IT	: Information Technologies
KPA	: Key Process Area
LA	: Largely Achieved
LOC	: Line of Code
M	: Model
MSF	: Microsoft Solutions Framework
PA	: Partially Achived
RAD	: Rapid Application Development
RUP	: Rational Unified Process
RQ	: Research Question
SDLC	: Software Development Life Cycle
SMM	: Scrum Maturity Model
SPICE	: Software Process Improvement and Capability Determination
TDD	: Test Driven Development
TFS	: Team Foundation Server
UML	: Unified Modeling Language
XP	: Extreme programming
WebRTC	: Web Real Time Communication
WP	: Work Products

CHAPTER 1

INTRODUCTION

Agile software development methods have proved their success since the publication of agile manifesto in 2001, and have gained significant acceptance by IT community by increasing business value, speeding up delivery, eliminating non-value adding activities to software development process and reducing the overall risk associated with software [1].

However, adopting agile methods is not easy or straightforward instead of common sense. [2] Agile concepts were extensively misinterpreted or "agile" was used as an excuse for being undisciplined by some of the organizations. [3]. As a result, organizations need assistance in adopting agile methodologies and identifying how far they are to be truly agile [4]. Structured approaches such as maturity models or frameworks aim to assist the transition of organizations to agile by providing comprehensive guidance on agile processes, introducing roadmaps and describing what it means to be "agile".

This thesis presents a structured Software Agility Assessment Reference Model (AgilityMod) developed with the aim of assessing agility level of software projects, identifying gaps that prevent to obtain maximum benefits from agile principles, and providing roadmaps on the way to be fully agile.

This chapter includes the specification of the background and the discussion of the problem. The purpose of the study is clarified before the discussion on the significance of the study. Next, research strategy to develop a Software Agility Assessment Reference Model are stated.

1.1. Background of the Problem

Traditional, plan-driven software development approaches rely on at least 60 years of assumptions about business, technology and organization structures [5]. They are obsolete and therefore counterproductive for rapidly changing business environments. Agile software development approaches are developed as a reaction to traditional methods that are characterized with extensive planning, heavyweight processes and bureaucracy [6]. Origins of agile software development approaches go back to early

1980s where like iterative, incremental and evolutionary software development emerged [7-10].

Agile software development methods are characterized with delivering working software to customer through short, time-boxed iterations and encouraging people to minimize bureaucracy, collaborating, self-organizing, embracing variability, balancing up-front work and just-in-time work, favoring adaptive and exploratory approaches and providing fast-feedback [11, 12].

The core set of Agile methods or Highsmith's phrase Agile Software Development Ecosytems (ASDEs) [13] include Dynamic Systems Development Method (DSDM) (1995) [14], Scrum (1995) [15], Agile Software Process Model (1997) [16], Crystal collection (1998-2004) [12, 17-19], Extreme programming (XP) (1999) [20, 21], Internet Speed Development (ISD) (1999)[22-24], Adaptive Software Development (ASD) [25], Pragmatic Programming (2000) [26], Feature Driven Development (2002) [27], Agile Modelling (2002) [28], Lean Software Development [29] and Test Driven Development (2003) [30].

The models which are developed for varying real life conditions share a set values which is later defined with agile manifesto in 2001 [31]. Agile manifesto which is signed by seventeen experts from different disciplines describes the following values:

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

This brief description of values inspired so many people, on the other hand, it created some significant misunderstanding in some software circles. The purpose of agile manifesto is to highlight the gap the between traditional approach and agile approach with strong terms. Doing agile does not mean to choose left side over right side, but rather maintaining the balance between two sides [1], [32], [33], [34].

Agile values described in manifesto are supported by twelve agile principles which are also published by members of Agile Alliance [31] . The principles, which can be found in Chapter 2, are like a bridge that connect the agile values and agile practices together. Principles and values construct a foundation of agile sense together and explain how agile practices work in practice [1].

1.2. Statement of the Problem

Agile software development methods are frequently adopted in the recent years by software community as they are seen as a complete solution for the problems like missing deadlines, exceeding budgets, delivering final products that do not meet the needs of the customer [1]. VersionOne presents in the state-of-agile survey that 52% of the projects are managed with agile techniques in software organizations [35].

Ambler explored how effective the five most common software paradigms: Lean, Agile, Iterative, Ad-Hoc and Traditional in the 2013 IT Project Success Rates Survey conducted in December 2013 [36]. The survey was performed with 173 participants. Lean

strategies such as Kanban are found most successful among other strategies for the delivery of the product and meeting projects' success criteria. Iterative and Agile strategies are followed that with a 5% percentage decrease. What is noticeable in these results is that agile and iterative strategies have approximately identical results with 64-65% success, 28-30% challenge and 6-7% failure rates. Similarly 2008 and 2011 IT Project Success survey found agile and iterative projects produced similar statistical results in terms of quality, success in deliveries, and return on investment [1].

As mentioned before, 30% and 6% of participants of the 2013 IT Project Success Rates Survey reported that they had experienced challenge and failure in an agile project respectively [36]. In the Agile Development Survey, 85% of the participants reported that they had experienced a kind of failure in agile project because of lack of cultural transition, lack of experience in agile methods and communication problems [35].

Ambler [1] also states there are an increasing numbers of project failures associated with agile strategies.

Both failure stories and identical success rates of agile and iterative software development projects indicate that organizations do not get a full benefit from agile software development techniques.

What we also observed from our personal experiences is that the organizations new at agile software development techniques start by selecting a few agile practices, adapting them in the way they prefer and convince themselves as doing agile software development until they see no improvement or even getting worse situation or "agile" is being used as an excuse for being undisciplined by some of the organizations.

Because of these reasons there is a fundamental need to assist organizations in adopting agile methods/practices and to guide them for improving their agile capability [3]. Structural approaches such as agile maturity/assessment models or frameworks aim to assist the transition of organizations to agile by providing comprehensive guidance on agile processes, introducing roadmaps and describing what it means to be "agile".

In the current state, there are about forty models related to agile maturity, including both academic publications and Internet publications [37, 38]. These models are grouped into three based on the classification of Schweigert et al.: ones that are influenced by the structure of CMMI, ones that have a specific leveling structure and ones that do not use explicit leveling structure [37]. They argue that these models do not measure the real agility and support guidance. Instead, they check for the implementation of some specific agile practices.

A more detailed discussion about current agile maturity/assessment models are given in the related research chapter.

In one of our previous studies [39] five of the most frequently referenced agile maturity models are applied in an organization and evaluated (Table 1). The evaluation is based on six quality criteria: fitness for purpose, completeness, definition of agile levels, objectivity, correctness and consistency.

Table 1 List of the Agile Maturity Models/Frameworks Evaluated

ID	Model Owner[20]	Name of the Model/Framework
M1	Patel and Ramachandran	Agile Maturity Model [40]
M2	Yin	Scrum Maturity Model [41]
M3	Sidky	Agile Adoption Framework [3]
M4	Benefield	Benefield's Model [42]
M5	Ambler	Agile Scaling Model [1]

The results of the study indicated that none of these models satisfies all the expected criteria and need to be improved in terms of scope, definitions of agility levels and objectivity. The most obvious deficiency of the models is that they do not support an agile process architecture holistically. Each model focus on different parts of the software development life cycle. None of the models has a well-defined structure with process inputs, practices and outputs forms.

Among this model quagmire, there is no commonly accepted agile maturity/assessment model. The need for a structured Software Agility Assessment Reference Model remains valid.

1.3. Purpose of the Study

The purpose of this study is to develop a structured Software Agility Assessment Reference Model (AgilityMod) to be utilized for the agility assessment of software projects and organizations. Such an assessment model shall enable the assessment artifacts to be utilized as a guideline for organizations to get better at agile values and principles through levels. We aimed AgilityMod is fully compatible with the agile process architecture (the structural design of the processes). The model shall provide a complete guidance so that organizations observe their weaknesses and problematic areas and implement the agile processes and practices correctly and in consistency with agile manifest. The model also shall provide means for helping them avoid incorrect tailoring.

We will be assessing the "agility", neither software process capability nor maturity. Erickson et al. [43] define agility as follows:

"Agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like."

From AgilityMod's perspective, "agility" is being able to give and obtain feedback rapidly, being adaptive to changing conditions, having a confidence on producing solutions to complex problems, being creative and innovative, respecting others, working with humility, learning from mistakes, improving continuously, solving problems/issues with communication and moving away from complex and bureaucratic procedures.

The model we developed ensures the achievement of these values level by level.

The need for structured models is accepted by whole IT community [38]. ISO/IEC 15504 (SPICE) and CMMI are the models which solve software process assessment model quagmire and brought standardization to Software Process Improvement initiatives.

AgilityMod's meta-model structure is defined in accordance with ISO/IEC 15504-Process Assessment Model. Our purpose of using ISO/IEC 15504's structure is to create a common basis for performing assessments of agility and present the assessment results using a common rating scale.

To briefly describe; the model mainly consists of two dimensions: Aspect dimension and Agility Dimension. In the aspect dimension, aspect purposes, aspect practices, outcomes, outputs and agile elaborations to aspect practices are defined. In the other dimension agility dimension, level of agility with agility indicators are defined. Agility indicators include aspect attributes, generic agility practices, generic resources and generic work products. Overall view of the Model can be seen in Figure 1.

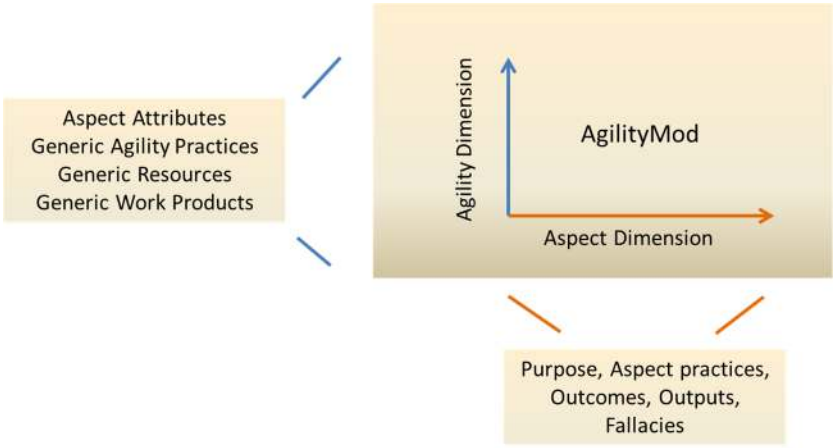

Figure 1 Structure of AgilityMod

1.4. Significance of the Study

Agile software development methods cover different phases of software development life cycle except for the DSDM (Dynamic Systems Development Method) and ISD (Internet Speed Development) [9]. They are collection of a group of practices that either focus on project management practices, technical practices or else. Before the adaptation of a specific agile method, its purpose and fitness to the organization's needs should be investigated. It is not suggested that organizations select and adapt a combination of practices from various agile methods especially if agile development is never used before [44]. In cases where this selection is performed, it is very possible left significant practices out that will provide a true solution to problems.

AgilityMod is developed being independent of any specific agile method. Its holistic structure enables the assessment of organizations being independent of the agile model used. Thus, the model can be used for capturing missing practices and proposing a complete and useful improvement solution to organization.

The model will also produce comparable results even for the organizations which use different types of agile software development methods.

Agile software development processes provide flexibility to project teams at the same time they require a certain type of discipline. However, it is observed that agile methods can be used as an excuse for developing undisciplined software development. From this perspective, AgilityMod, with its all practices, outcomes, outputs and resources, defines the edges of agile software development and supports correct tailoring of agile principles and values. AgilityMod will provide an objective assessment for the agility level of projects and/or organizations. Thus differentiation of the disciplined agile organizations and ad-hoc organization is possible.

Defining AgilityMod being compatible to ISO/IEC 15504 – Process Assessment Model (SPICE) will enhance the applicability of the Model. ISO/IEC 15504 does not present a specific agile tailoring for software processes in its latest version and the Model is subject to improvement in that area. An enhancement of SPICE from agile perspective is very possible in the close future. Through our relations with ISO/IEC 15504 community, we aim to initiate studies to transform AgilityMod into an international standard.

1.5. Research Strategy

The research strategy followed through this thesis study is given step by step in Figure 2. The study is performed in the nature of the "qualitative research". Descriptions given by Creswell [45] justify the selection. He mentions that; in qualitative research, researchers collect data in the natural settings through the overview of the documents, observing the behavior or interviewing the participants. Data can be collected from multiple sources and the research process flows from forth to back and back to forth until a comprehensive model is developed [45].

Before the development of the assessment model, we performed a literature survey on current agile maturity and assessment models and selected a set of models to analyze their structure deeper. Then, we observed the selected agile maturity models' applicability, strengths and weaknesses with a multiple case study in a software organization. The literature review is also conducted on agile software development methods to specify their common and specific characteristics and nature.

Based on the findings of the first case study, we identified the requirements and essential characteristics of an agile assessment model and developed the first version of the model, AgilityMod. The model went through a two-phased refinement process to reach a ready to be applied maturity. The first refinement is performed after the direct observation of the applicability of the model in the field through a case study. The second refinement is performed based on the feedbacks of the agile and process assessment experts and practitioners. In this scope, the model is reviewed by a CMMI lead assessor from India, a process assessment consultant from Turkey and an agile practitioner and lecturer from Canada.

The validation of the Model are achieved through the implementation of AgilityMod in six software organizations in the scope of a multiple case study. We conducted formal assessments through semi-structured interviews with process practitioners, and evaluate direct evidences. We analyzed the assessment process and present the result of

each assessment as report. Over the reports, we discussed the results with practitioners and asked if the results correctly represent the agile state of the projects/organization.

In the last step, we answered the following research questions in the light of the case studies:

RQ1: How suitable is the third version of Software Agility Assessment Reference Model to be used with the purpose of identifying aspects' agility, identifying the agility gaps and providing roadmaps for improving in agility in a software project?

RQ2: What are the strengths and weaknesses of the third version of AgilityMod?

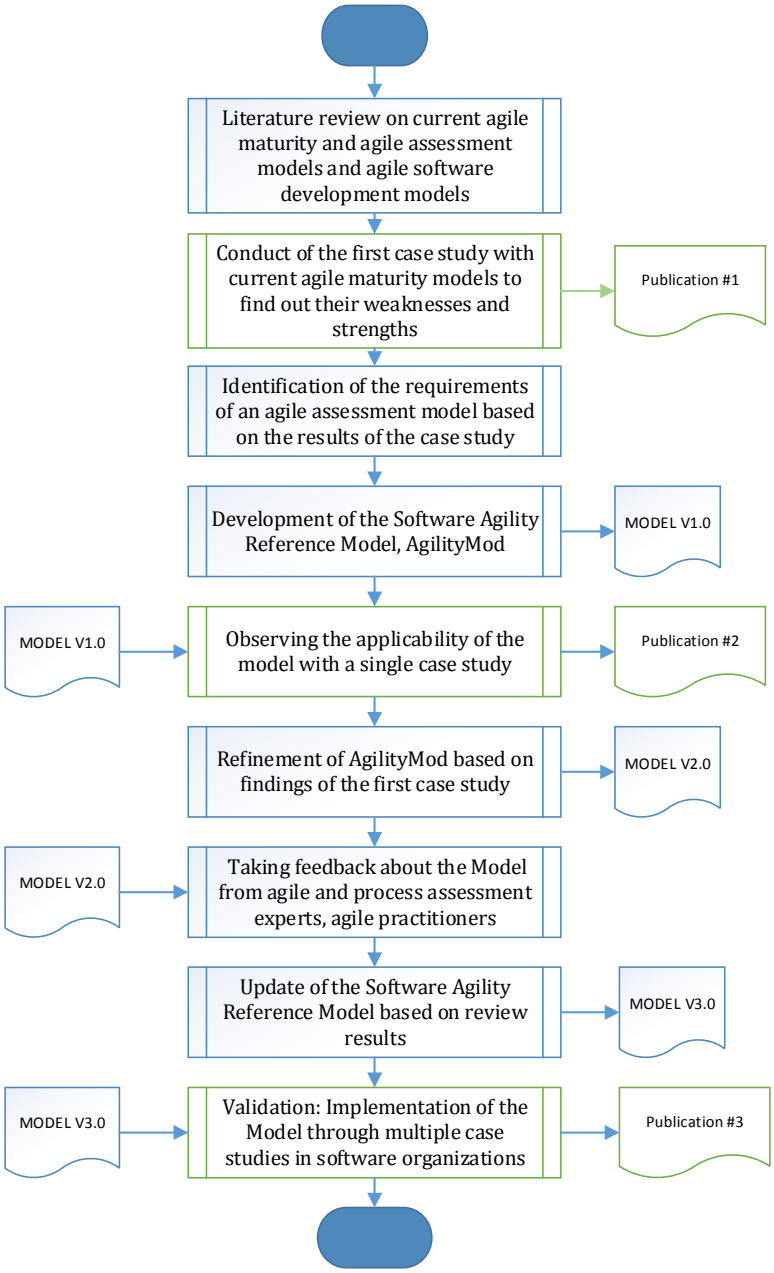

Figure 2 Steps of the Research Strategy

1.6. Organization of the Thesis

The rest of this thesis is organized as follows:

Chapter two is a review of the literature in agile software development methods, agile maturity and assessment models. In this chapter we explain the case study that we conducted to observe the usability of current agile maturity models, their strengths and weaknesses.

Chapter three describes the structure and components of Software Agility Assessment Reference Model (AgilityMod) we propose in this study. Detailed description of aspect and agility dimensions are provided in Appendix A.

Chapter four describes both the exploratory case study and multiple case study conducted with exploration and validation purposes.

Chapter five describes the overall findings, achievements and future work.

Appendix B provides the agility assessment report developed based on AgilityMod_v3.0 of Case 1 as an example.

CHAPTER 2

RELATED RESEARCH

The purpose of this chapter is to review the literature to identify the agile software development methods and agile maturity models. It is important for the purpose of the study to understand the structures of agile software development methods to get insight on the common and different characteristics of the methods. By understanding existing agile software development methods, it is possible to develop a holistic Software Agility Assessment Reference Model covering all the methods like an umbrella. This chapter also includes review of research on existing agile maturity models. We paid special attention to understanding the structures of the models as well as their weaknesses and strengths. Among forty agile maturity models, we evaluated the applicability and sufficiency of most referenced ones through a multiple case study.

Section 2.1 provides information about brief history of agile software development methods. Section 2.2 explains the characteristics of the agile software development methods that construct the baseline for the structure of AgilityMod. In Section 2.3, existing agile maturity models that have academic point of view are specified and usage of these models in research studies are focused. The structure of the AgilityMod is influenced by the meta-model of the ISO/IEC 15504-Process Assessment Model. In Section 2.4 we briefly describe the structure of ISO/IEC 15504 and the reasoning behind the selection of this standard as a basis.

2.1 Agile Software Development

Agile software development is one of the most important paradigms that changed the way of developing software radically. Debates on what agile is still continues if agile is a development and management philosophy, a collection of technical practices, a way of life or all them [6].

Agile's meaning is *"able to move quickly and easily, nimble and dexterous*" based on Oxford English Dictionary [46]. The word "agile" was used with "software development" for the first time in the study of Aoyama in 1997 [16]. The introduction of extreme programming has been accepted as the starting point for various methods. [47]

Nerur et al. specifies that agile and traditional approaches diverge in a number of aspects: approach to control, management style, knowledge management, role of the customer in development process, role assignment, communication style, development life-cycle,

organizational culture and technology [48]. Boehm makes this discussion over developers, customers, requirements, architecture, refactoring, team size and primary objective of development [49].

Agile methods are characterized with contributing the creation of change, being provocative in advance of change and learning from change rather than rejecting and taking precautions to prevent it [50]. Highsmith states that agile software development is about knowing how to balance structure and flexibility [51]. He further underlines that "agile" is more than lightweight processes, less ceremony and fewer documents. What keys to agile ecosystems are focusing people factor, giving them the power to make quick decisions, self-adapting and improving their own processes [51].

Stober and Hansmann resemble the characteristics of software development teams to characteristics of fractal units in mathematics which are self-similarity, goal-orientation, self-organization, self-improvement and vitality [52] .

From 1990s to early 2000s agile phenomenon is shown itself as increase in the numbers of models. Historical roots of agile software development go back to 1980s. The origins of these models and the models itself are shown in Figure 3 .

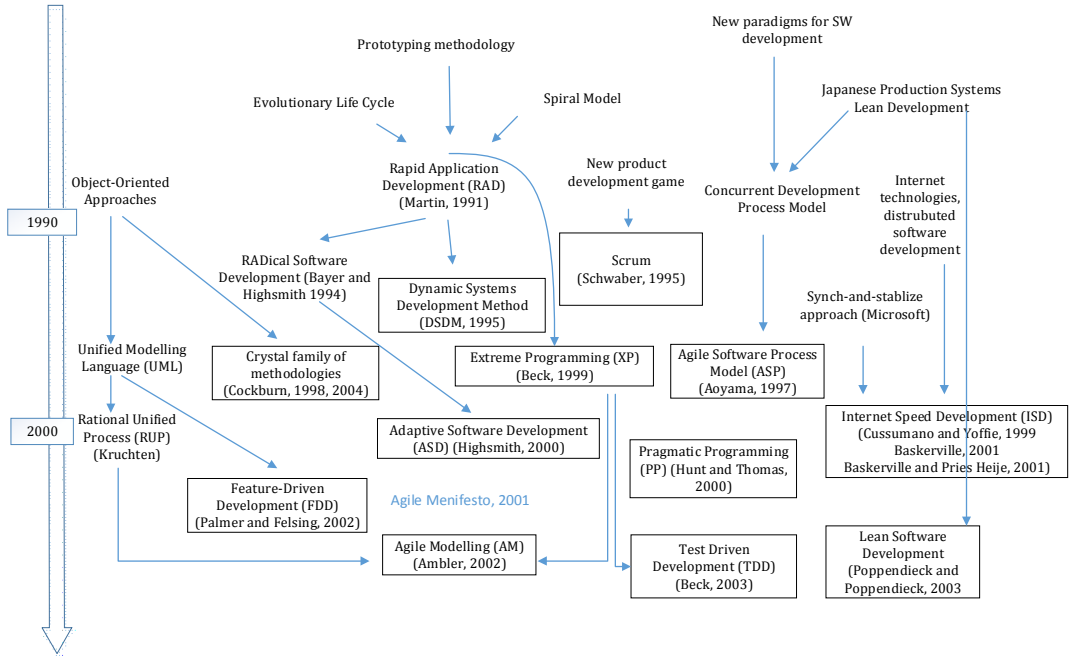

Figure 3 Origins of Agile Software Development Methods –adapted from [9].

2.1.1. Agile Manifesto and Principles

In 2001, 17 people who contributed the development of XP [20, 21], Scrum [15], DSDM [14], ASD [25], Crystal [12, 17-19] and pragmatic programming [26], signed Agile Manifesto [53] [31]. We specified the manifesto items in Chapter 1.1. Here we briefly explain them to clarify any misunderstanding about the values:

Individuals and interactions over processes and tools: Tools and processes are important and have positive effects on software efficiency. However, what more valuable is the interactions between well-skilled team members and creating such a communication environment for them [1]. What guarantees the success is not process descriptions, templates or guidelines but skilled, self-organizing, self-motivated and trusted people and how they work together.

Working software over comprehensive documentation: Primary goal of doing all the work is developing a working solution. Intermediate document artifacts provide no customer value and subject to continuous updates through the development. The purpose is to avoid non-value added or unnecessary documentation prefer not to develop unless it is valuable to customer [11].

Customer collaboration over contract negotiation: This value focuses on the significance of working closely with the customer. A contract even if it is necessary to have, cannot be a substitute of effective communication [54]. Teams should spend their effort to discover what the customer wants. Even if the emphasis is the "customer", there will be need to interact more than business customers (i.e other stakeholders) to understand the true needs [1].

Responding to change over following a plan: This value focus on the adaptability capability of the teams to changes in software development process. Change is inevitable and all software development process should be lean and simple enough to adapt the upcoming changes but this does not mean not to have a project plan. Rather it should be detailed enough.

What inspiring so many people is not just agile manifesto but also the agile principles which brings explanation to manifesto items [31]. We list the principles below:

Table 2 Agile Principles

No	Agile Principle
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4	Business people and developers must work together daily throughout the project.
5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7	Working software is the primary measure of progress.
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9	Continuous attention to technical excellence and good design enhances agility.
10	Simplicity--the art of maximizing the amount of work not done--is essential.

11	The best architectures, requirements, and designs emerge from self-organizing teams.
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Adapting agile software development models requires working conformance to these principles.

2.1.2. Agile Software Development Methods

Abrahamsson et al. discuss what makes a method agile. Beyond sharing values in agile manifesto, agile methods propose an iterative and incremental development in a cooperative environment. They are easy to learn and modifiable as well as being adaptive to changes [47]. In this sub-chapter we briefly describe the characteristics of major agile software development methods.

Dynamic Systems Development Method (DSDM): DSDM is specified as the first truly agile software development method [55] [56]. Origins of DSDM goes back to early 1990s when the Rapid Application Development (RAD) [57] is spoken in software circles in United Kingdom. It was developed by a non-profit organization, DSDM Consortium who are also co-authors of Agile Manifesto. Unlike traditional approaches, DSDM fixes the time and the cost and adjusts the functionality by keeping the quality in a desired level [14]. The model defines processes, people, products and practices and the ideology of DSDM through principles and philosophy. However, no detailed specifications are provided for practices by mentioning that each organization is different [14]. By means of that the model allows adaptation.

Among all other agile software development methods, DSDM delivers a solution for the whole software development life cycle by providing support from project inception to release of a system [56]. Abrahamsson *et al.* specify that although the DSDM Consortium is aimed to develop a public model, no empirical and scientific results discussing the validity of the model are provided [56]

Extreme Programming (XP): Extreme Programming, was developed by Kent Beck in 1999, provides a collection of software engineering practices [20, 21]. Even though the practices are not novel, what XP achieves is bringing them together to function for being adaptive to changes and to produce quality software at a sustainable pace. XP defines values, principles and roles. Some of the fundamental practices of XP are planning game, small releases, metaphors, simple design, continuous unit testing, refactoring, pair programming, collective code ownership, continuous integration, work 40-hour-a-week, on-site customer and coding standards [21]. By basically focusing on technical practices, XP does not provide support for project management except for the adaptive planning [56].

Scrum: Scrum was developed by Schwaber with the purpose of providing a management framework for software development [15]. The fundamental idea behind Scrum is to apply process control theory to software development to achieve flexibility, adaptability and productivity [47]. It relies on a set of values, principles and practices which can be adopted based on specific conditions. Scrum gives value on providing frequent feedback, embracing and leveraging variability, being adaptive, balancing upfront and just-in-time

work, continuous learning, value-centric delivery and employing sufficient ceremony [11]. It offers effective solutions by providing specific roles, artifacts, activities and rules.

Scrum does not provide any specific engineering practice for implementation. Schwaber and Beedle suggest implementation of other agile methods to complement Scrum as a complete software development approach [58].

Agile Modeling (AM): Deciding sufficient level documentation is one of challenging tasks in agile software development. Agile Modelling (AM) developed by Ambler, aims to provide a framework for effective modeling and documentation of software systems [59].

Even though the other agile software development methods like XP, Scrum or DSDM do not abandon modeling, AM can be considered as a complementary method for a complete solution. Ambler proposes core and supplementary practices for effective modeling. Some of the fundamental practices include iterative and incremental modelling, modelling as a teamwork where stakeholders are involved, collective ownership of models, simplicity, proving the models with code, and modelling considering testability [59].

Crystal family: Crystal family is a collection of methodologies each of which is published as a book [19]. Cockburn who is the founder of crystal family describes crystal family as a collection of lightweight methodologies [17]. Each method is developed as a solution for software organizations in different sizes and criticality. Crystal collection focuses on cooperative work, communication, interaction and improving people's skills. Each method is specified with different color codes indicating the needs of different strategies of organizations. There exist eight different types of crystal method. Among them, Crystal Clear and Crystal Yellow are suitable for small teams whereas Crystal Diamond and Crystal Sapphire are suitable for organizations more than 200 people. Suggested roles and practices changes for each crystal method which makes each method scalable.

According to Abrahamsson's [56] analysis crystal methods delivers solutions for only design, coding, and testing phases of a software development life cycle .

Rational Unified Process (RUP): RUP, which is a well-defined software development process, was developed by Rational Software Company in the mid 1990s [60]. RUP is an iterative approach for object-oriented systems and provides a customizable process framework [47]. Some of the researchers define RUP as one of the agile software development methods [52] whereas others leaves out of scope in agile methods' classification and see as a pioneer of agile methods [56]. RUP divides software development process into 4 phases including: inception, elaboration, construction and transition. Implementation starts at the inception phase fast feedback is provided with 2-weeks cycles. RUP also shares some of other agile values and principles like: effective modelling through Unified Modelling Language (UML), working closely with customer, delivering value, focusing on working software, creating an executable architecture early and working together as team [60].

Feature-Driven Development (FDD): FDD [27] was created by Peter Coad, Jeff De Luca and Stephen Palmer as a process-oriented software development method for developing large-scale software projects. It combines the practices of agile software development and model-driven development. What makes unique FDD is its suitability of developing

critical software systems. FDD's major focuses on software development life cycle are design and building phases [56]. Plans and design are performed by feature. Iterations aim to deliver working features in one to three-week time period.

Adaptive Software Development (ASD): ASD was developed by James A. Highsmith as solution for development of large and complex systems [56]. ASD's origins go back to 1994 when RADical Software Development is published by Bayer and Highsmith [25, 61]. ASD relies on iterative and incremental software development. The method consists of three life cycle phases: Speculation, Collaboration and Learning which refer to planning, concurrent development and improvement phases respectively [25]. ASD is not a task-oriented approach that means working product is more important than the process itself [47]. Highsmith also make emphasis on the adaptation of the organization' culture besides other adoptions. The principles in ASD are very similar to the principles of other agile methods like iterative and time-boxed development, being change tolerant and risk-driven.

Lean Software Development: Lean Software Development is an adaptation of Toyota Production System's lean thinking principles to software development [29]. The approach relies on seven principles: eliminating waste (excessive ceremony and project management), continuous learning, building quality in, deferring commitment, delivering fast, respecting people and optimizing to improve the value stream.

In the literature there exists studies that analyze agile software development models from different perspectives. Abrahamsson et al. perform a comparative review of agile models from the following perspectives: project management support, life-cycle coverage, type of practical guidance, adaptability in actual use, type of research objectives and existence of empirical evidence [56]. The results of the study indicate that without giving any rationale models focus on different phases of software development life cycle. Some of the models (Agile Modelling, Extreme Programming and Pair Programming) do not provide a project management support. On the other hand, only Agile Modelling, Extreme Programming and Pair Programming provide guidance on how to adopt suggested solutions [56].

In 2008, Dyba and Dingsoyr performs a systematic review on empirical studies of agile software development [62]. They investigate benefits and limitations of agile methods through 36 empirical studies. Study reveals that XP is the most evaluated method on empirical studies, and there is still need to evaluate other agile models through empirical studies.

2.2. **Agile Maturity and Assessment Models**

In this chapter we are going to discuss about the status of current agile maturity and assessment models and describe their nature. Table 3 below lists all the maturity models found in a comprehensive search conducted both on scientific research platforms and web. Our studies indicated that there is a model quagmire in the field.

What we observe during the search is that the models published with assessment purpose extensively use the "maturity" keyword. Below we use maturity keyword interchangeably for the models developed with "assessment" purpose.

Schweigert et al. mention that there are approximately 40 agile maturity models published in various mediums [37]. We list 15 models which comply with our minimum expectations from a publication. We included all the agile maturity models published in a book, a journal or a conference proceeding. Elimination is required for the resources generally published in personal blogs of agile practitioners. We excluded the "models" which only consists of brief descriptions where the blog was used as a brainstorming area of the author. We also excluded the publications which only deliver key questions or recommendations about agile adoption.

Table 3 Agile Maturity Models exist in the Literature and Web

ID	Name of the Model	Model Owner
M1	Agile Maturity Model [40]	Patel and Ramachandran
M2	Scrum Maturity Model [41]	Yin
M3	Agile Adoption Framework [3]	Sidky
M4	Benefield's Model [42]	Benefield
M5	Agile Scaling Model [1]	Ambler
M6	Agile Maturity Model [63]	Humble and Russel
M7	Simple Life Cycle Agile Maturity Model [64]	Malic
M8	Agile Maturity Model [65]	Proulx
M9	Agile Maturity Model [66]	Jayaraj
M10	The Agile Maturity Model [67]	Ambler
M11	Agile Maturity Model [68]	Anderson
M12	Agile Maturity Model [69]	Banerjee
M13	The Maturity Curve [70]	Bavani
M14	Agile Testing Maturity Model [71]	Ronen
M15	An Agile BI Maturity Model [72]	Woods

2.2.1. Case Study to Evaluate Current Models

A subset of the models listed above is subject to a deeper analysis. We planned to conduct a multiple case study by utilizing the selected models in a software organization for agility assessment. The results of this study is also published in Software Process Improvement and Capability Determination (SPICE'13) conference [39]. We aimed to answer following research questions (RQ):

RQ1: How sufficient are the existing agile maturity models in providing insight about an organization's agile capability?

RQ 2: What are the strengths and weaknesses of the agile maturity models?

2.2.1.1. Design of the Case Study

In order to ensure the objectivity and correctness of the evaluation process, we decided to perform the assessment based on a set of evaluation criteria which had been previously utilized in similar studies. We planned to review the literature to determine those criteria. To answer the research questions, we planned to select a software development organization, which develops information systems projects, and claims to apply agile practices/processes within one year time at least in more than one project. We planned to conduct gap analyzes to apply the models and to determine agile maturity of the organization relative to reference models. The major aim of the practical assessment is not directly to identify the organization's agile maturity, but to observe the

applicability of the models, and to identify the strengths and weaknesses of them with hands-on practices.

2.2.1.2. **Conduct of the Case Study**

Among 15 models, the first 5 one were selected to be included in the scope of detailed analysis. The inclusion criteria were determined as follows:

1) Detailed description of the model should be available for detailed analysis.
2) The study should have been published in one of the major conference proceedings, journals or books, which is an indicator of academic perspective of the model.

M1-M2 and M3 complied with the first criterion and M1-M2-M3-M4 and M5 complied with the second criterion. We took the union of these results and evaluated five models even if we don't have detailed descriptions of M4 and M5.

The Case*:* The organization that we conducted the case study is developing various management information systems related with the digitization of the procurement procedures, health management and law tracking systems. It is a small sized company with sixty employees. The organization is found appropriate for the case study since agile processes have been applied for 1.5 years in small or medium-scaled software development projects.

Selection of Evaluation Criteria: Although there has not been such a study which assessed the qualification of the agile maturity models and published assessment criteria, we examined similar studies in the literature performed with CMMI or ISO 15504 to identify assessment criteria. Rout et al. criticize "the purpose, the scope, the elements and the indicators" of CMMI and mapping capability of CMMI with ISO 15504 and maturity results' verifiability based on completeness-clearness-unambiguity criteria. ISO/IEC 15504 Part7- "Assessment of Organizational Maturity" defines the purpose of this part as ensuring that the assessment results are objective, impartial, consistent, repeatable, comparable and representative of the assessed organizational units [73]. In his book Kneuper, assesses the limitations of CMMI in terms of definition of maturity levels and completeness of processes [74]. He also examines CMMI and product quality relation and minimum size of organizations suitable to use CMMI.

We set the following assessment criteria being compatible with those studies above:

Fitness for Purpose: An agile maturity/assessment model must be developed with the purpose of assessing agile process capability and assisting organizations in software process improvement.

Completeness: An agile maturity/assessment must address all or a subset of major engineering and management processes within a software development life cycle. It must include process related definitions, goals, practices or process success indicators which enable assessment of the agile processes.

Definition of Agile Levels: An agile maturity/assessment model must provide definitions of agile levels which enumerate the different degrees of agility. Those maturity levels could be interpreted intuitively and must be designed to complement each other.

Objectivity: Verifiable results must be produced. The judgment of the assessor must be at a minimum level.

Correctness: All model elements must be compatible with agile principles. Descriptions, goals and work products must correctly represent the related process or process area.

Consistency: An agile maturity model/framework must be internally consistent. All processes and practices must be at the same logical level. There mustn't be logical or temporal conflicts between two specified model elements.

Conduct of the Gap Analysis: We performed five separate gap analysis study using the first five maturity models/frameworks (M1-M2-M3-M4 and M5) listed in Table 3. The major purposes of these gap analyzes were to identify weaknesses and strengths of the models and their usability/applicability, while assessing the organization's software development processes.

Author of the thesis performed assessment meetings with the project manager and quality manager who had involved the management of various agile software development projects in the organization.

At the beginning of the gap analysis, we obtained the general overview of the processes and the team structure of the organization Then, we asked specific questions based on the specified goals, processes, practices and example work products of the models to understand if the requirements of each model are achieved in the projects, or not. In the cases of M1, M2 and M3 the assessment questions had already been provided in the model.

The results of the gap analysis were reviewed by the team who involved in the assessment. Fuzzy issues were clarified and corrections were made at this phase.

The duration that was required to perform a gap analysis was determined by the detail level of the model, and it got shortened towards the end of the analyses, since we had already known most of the answers. The gap analyzes were finished in 6 person/days in the organization.

In the following days, the models were examined in more detail considering the notes that we had taken during the gap analyzes. Based on the results of the gap analyzes, the organization's agile maturity levels are presented in Table 4.

Table 4 Organization's Agile Maturity Based on Five Models

Maturity Models	**MI**	**M2**	**M 3**	**M4**	**M5**
Maturity Levels	Level 1	Level 2	Level 0	Level 0	Not Agile
Name of the Level	Initial	Managed	Not Exists	Not Exists	Not

According to the assessment we performed with Scrum Maturity Model (M2), the organization's maturity level was determined as Level 2: Managed. It was observed that technical agile practices such as test driven development or continuous integration were not carried out in the organization, but agile practices for project management and project tracking activities were performed. Because of this reason, the maturity level of

the organization was determined as Level 2: Managed with Scrum Maturity Model. However, the organization does not satisfy the first maturity level requirements of the M3, M4 and M5 models. According to the assessment results performed with the models that evaluate the agile processes also in technical aspects (M1, M3, M4, M5), the agile maturity of the organization was determined as Level 1 or Level 0.

2.2.1.3. **Findings of the Case Study**

In this subsection we present the strengths and weaknesses of the models identified during or after the case study. M1, M2 and M3 were analyzed in more detail compared to the other models since there exists comprehensive references such as thesis or journal papers about them.

M1, Agile Maturity Model: "Agile Maturity Model" [40] was developed by Pathel and Ramachandran with a similar structure to the CMMI. It defines the agile maturity in five levels from "initial" to "sustained".

The model has been evaluated based on the parameters given in Section 2.2.1.3.

Fitness for Purpose: The model has been developed with the purpose of enhancing the adaptability of agile software development methodology and its practices and providing both a software process improvement (SPI) framework and a maturity assessment framework. Although the adequacy of them is questionable, for each maturity level, key process areas and related assessment questionnaires were defined to enable SPI and maturity assessment.

Completeness: For the key process areas (KPA) in each maturity level; process descriptions, goals and example work products, were not explicitly defined. Therefore the analysis has been conducted based on the descriptions in the questionnaires instead of process goals and practices.

Although the model includes many processes, it doesn't cover all of the software engineering processes, such as configuration management, change management and project monitoring and control.

There is no distinction between the optional and the mandatory practices of the processes.

Definition of Agile Levels: The maturity levels do not complement each other in providing a combined benefit. Gap analyses results showed that, the organization is more successful in achieving the process requirements of Level 4-"Improved" than the process requirements of Level2-"Explored" and Level 3-"Defined" levels. This indicated that the KPAs of 4th Level, which are; project management, sustainable pace, risk assessment and self-organizing team, could have been considered in previous levels.

Objectivity: The model uses a subjective language such as "The customer relationship is maintained very well at this level". As the process definitions include subjective words such as "very good", "better", objective assessment and identification of process improvement goals were not possible.

Correctness: Most of the assessment questionnaires, which were defined for each KPA in each maturity level, consist of the questions/descriptions with no direct relation to the KPA. For example, for the KPA of "On-site customer availability" there exists a description in the questionnaire such that "there is a plan exists to manage story cards", which should obviously be considered in Project Management KPA. There is not a direct relation between the KPA of "Delivering Working Products/SW Frequently" and the description of "Only one pair integrates the code at a time". The "story card driven development" KPA includes questions/descriptions related with defect prevention and detection.

Although the goals of 5th maturity level were determined as "tuning project performance" and "defect prevention", KPA's of this level were set as "project planning" and "story cards driven development", which had no relation with these goals. That means it is impossible to meet maturity level goals with corresponding KPAs.

Consistency: Consistency among the abstraction levels of KPAs is weak. Some of them are at process level while the others are at practice level. For example, one KPA from Level 2 is "Project Planning" and the other one is "On-site Customer Availability". "Coding Standards" is a KPA for Level 3; however, it can only be a part of higher process such as development.

Questionnaire is the only part that enables detailed analysis of the model. However, contrary to the conventional structures of questionnaires, it includes sentences of order such as "obtain commitment to story cards"; flat (regular) sentences such as "customer is always available".

The existence of spelling and grammar errors and internal inconsistencies significantly decreases the readability. For example, the name of the 5th Level is "Sustained" in one section; and "Mature" in one of the following sections. In addition, the formula given for assessment and the example given to explain the related formula is not consistent.

Some of the KPA names do not coincide with the descriptions in the related questionnaire. For example, Risk Assessment KPA also includes questions about risk management.

M2, Scrum Maturity Model: Scrum Maturity Model [41] was developed to validate and improve Scrum based software development processes by Yin in 2011.

Fitness for Purpose: The scope of the model is limited with Scrum. It provides a mechanism (questionnaires for each maturity level) for the assessment of organizations' Scrum maturity in terms of Scrum practices. However, there is no defined procedure to decide whether the outcome is failure or success once the questionnaires are completed.

The model could be suitable for the improvement of Scrum practices following the set goals for practices, but not for the agile processes. Due to these limitations the model does not fully meets the requirements of "fitness for purpose" criteria.

Completeness: The model includes seven "goals", which refers to the processes or the process areas. These are; Basic Scrum Management, Software Requirements Engineering, Customer Relationships Management, Iteration Management, Standardized

Project Management, Measurement Analysis, and Performance Management. Other major processes such as testing or configurations management are not covered in the model. This is probably because Scrum did not specifically define these processes.

The questionnaire to assess compatibility of an organization to maturity level 4 includes just one question, which leaves other aspects such as quantitative project management and measurement and analysis out of the assessment context.

Definition of Agile Levels: The model includes five maturity levels with a similar structure to CMMI. However, when the density of goals and practices for each maturity level and the capability of the organizations to perform these practices were examined, it was observed that it would be more rational to describe Scrum maturity with fewer numbers of levels. The third level contains objectives such as "existence of definition of done and product owner", "planning iterations" and "conducting sprint review meetings" however, these are fundamental scrum practices that could be included in second level. Practices and objectives defined in the third level could not move an organization to an upper level.

Objectivity: The definitions of goals and practices are written in an objective language. However, there is no defined procedure for the assessment of the questionnaire results. The relation between the achievement of a maturity level and the amount of successfully answered questions is not clear.

Correctness: The model was developed based on the rules of Scrum methodology. From the Scrum perspective, there is no issue disrupting the correctness criteria in the model.

Consistency: There is no evident internal or external inconsistency issue in the model.

M3, Agile Adoption Framework: The Agile Adoption Framework (AAF) [3, 75] has been developed by Sidky in 2007. It includes two components; a measurement index for estimating agile potential and a 4-Stage process improvement process inspired from Deming Cycle.

We assessed the AAF based on 6 criteria determined at the design phase of this case study. We present the assessment results below:

Fitness for Purpose: The framework has been designed to enable the assessment of agile practices at project level and organizational level, and to provide guidance for organizations to adopt agile practices. It defines a roadmap for the agile adoption. However, the framework does not cover agile best practices to highlight how to overcome the weaknesses, which are essential for software process improvement. Due to these properties, the framework "largely" meets the "fitness for purpose" requirement of an agile maturity framework.

Completeness: Each agile level consists of a cluster of agile practices which were classified based on five agile principles. Actually, those five principles capture the essence of the whole 12 agile principles published in agile manifesto. However, in the further phases, the customer collaboration principle has been left out of the scope of the study, which caused the framework to lack one of the major agile principles.

The coverage of all processes is not in sufficient detail as in the case of configuration management process. It has only been defined with the existence of configuration management tools.

In addition, the framework does not include best practices, which guides the business in software process improvement, and highlights how to overcome process weaknesses. Another model is needed to complement the AAF in this respect.

Definition of Agile Levels: The framework defines agility in five levels, which has designed to cover agile values in agile manifesto. The reasons behind the order of the agile levels and, directing the organization to move toward agility were described in detail.

Objectivity: Project level and organizational level assessments are performed based on the questionnaires defined in AAF. Once the questionnaires are filled, the results are assessed based on a mathematical "evaluation methodology". However, questions are answered with interview or observation techniques which may cause subjective results. Therefore, objectivity criterion is not fully achieved.

Correctness: The framework is compatible with agile principles and agile manifesto. Process indicators are correctly identified.

Consistency: The name of the 5th Agile Level was specified as "Ambient " in [75] and "Encompassing" in [3] respectively. There is no other internal or external inconsistency in the framework.

M4, Benfields' Model: The resource available for Benefield's model is limited to a single published paper [42]. The model contains 5 levels of agile maturity (Level 1: Emergent Engineering Best Practices-Level 5: On Demand Just In Time Releases).

Benefield defines agile maturity with seven dimensions: Automated Regression Testing, Code Quality Metrics, Automated Deployment and Backout, Automated Builds and Configuration, Management best practices, Interlocked Delivery and Interface Integration Testing, Test Driven Development (TDD), Performance and Scalability Testing.

It is the strength of the model that Benefield took into consideration not only the managerial aspects of agile, but also the technical perspectives such as automated deployment, automated builds as an essential part of agile maturity in his model.

Fitness for Purpose: The model was developed to assess agile maturity and identify targets to improve agile maturity.

Completeness: The model focuses only on the agile practices given above instead of all the agile processes within a software life cycle. Major processes such as project planning, project management, project monitoring and control, change management, were not handled in the model.

The model does not include any evidence about how an agile practice is successfully achieved. Although the high-level goals and practices for each maturity level exist, detailed characteristics or practice based goals are not defined.

It is not possible to analyze the other assessment parameters; Definition of Agile Levels, Objectivity, Correctness and Consistency since the reference material does not include necessary information such as description of practices, objectives to achieve each dimension.

M5, Agile Scaling Model: Agile Scaling Model (ASM) [76] was developed by Ambler from IBM. It is a framework characterized by three levels. The first level is the application of core agile development methods in the organization, such as Scrum or extreme programming. In the second level, the focus of the organization is not only the development processes, but also the full agile delivery from project initiation to project closure. In the third level, disciplined agile delivery is applied in accordance with eight scaling factors covering the range of complexities that a team faces, such as large development team, or geographic distribution.

Although the ASM has not been referred as a maturity model, it presents a roadmap for the adoption and tailoring of the agile practices. Therefore, we evaluated ASM, based on predefined assessment criteria to identify to what extend it is to be used in improving agile maturity.

Fitness for Purpose: The structure of the model is not suitable to assess the agility level of the software development processes in an organization. It couldn't be used for an assessment, since it does not describe process related practices, goals, or any assessment questions.

Completeness: ASM does not prescribe how to successfully achieve the 1st scaling level, core agile development methods. It does not describe which practices to apply at which level, and does not focus on any process or practice descriptions. However, the 2nd Level is explained in detail by Ambler in his book [1]. The 2nd Level, Disciplined Agile Delivery, can be considered as a standalone software development life cycle (SDLC). The model is presenting various agile practice options for each phase in SDLC; however, it does not provide guidance on agile process assessment.

Description of the Agile Levels: The model requires the full application of one of the core agile development methods (i.e. Scrum or Agile Modeling) in the 1st Scaling Level and expanding the adoption of the agile practices to whole project life cycle, from initiation to closure, in the 2nd Scaling Level. That means; all the software development processes should be fully achieved in the first and second scaling levels. However, in reality, organizations gain process capabilities in an evolutionary way. Each scaling level could be divided into sub-levels to enable the improvement with small steps and to observe the progress in agile processes more clearly.

The reference material [76] has been written with an objective language. It is consistent with agile principles and there is no internal or external consistency problem. However, the reference material does not cover all the details of the ASM. Therefore, objectivity, correctness and consistency criteria could not be assessed fully.

To sum up, we assessed the characteristics of five agile maturity models/frameworks from software process improvement and process assessment perspectives, and identified their strengths and weaknesses by conducting a multiple case study.

Except from SMM [41], the models/frameworks are independent of any particular agile method. Agile maturity has been described through the agile processes or agile practices in the models/frameworks. As a result of the case study, we have found deficiencies in all of the models/frameworks at a certain level, according to six assessment criteria (fitness for purpose, completeness, definition of agile levels, objectivity, correctness and consistency). Table 5 depicts the results. We used a four-level scale to express models' qualifications relative to each other: "Not Achieved"- "Partially Achieved (PA)"- "Largely Achieved (LA)"- "Fully Achieved (FA)".

Table 5 An Overview of Assessment Results

Criteria / Models	Fitness for Purpose	Completeness	Definition of A. Levels	Objectivity	Correctness	Consistency
M1	**FA**	PA	Not Achieved	LA	PA	Not Achieved
M2 (SMM)	LA	PA	PA	PA	**FA**	**FA**
M3 (AAF)	LA	PA	**FA**	LA	**FA**	**FA**
M4	LA	PA	Not Applicable	Not Applicable	Not Applicable	Not Applicable
M5 (ASM)	PA	PA	PA	Not Applicable	Not Applicable	Not Applicable

Among all models/frameworks, AAF [75] has obtained the best assessment results. Its well-defined structure could be extended to cover agile best practices. SMM [41] is in the second rank following AAF. SMM's fundamental problems are not covering the major processes and urging to identify the Scrum maturity in 5 levels.

We couldn't found necessary information in the available references in the literature for a complete assessment of BM and ASM. However; the findings of the case study revealed that these models need to extend their agile coverage and improve the way of describing "how to be agile". The last model, AMM needs significant improvement in terms of definition of model elements, correctness, consistency and coverage.

This case study has underlined the observation that there is a need to improve the maturity models for better guidance in agile process adoption, process improvement and process assessment.

2.3. **Current Discussion on Agile Maturity Models**

Characteristics and quality of current agile maturity/assessment models are also evaluated by other researchers. Schweigert et al. perform a study to compile current available maturity models [37, 38]. They keep the scope broad including web resources or non-academic publications. They question the models from a different perspective than we described above. They evaluate if a commonly accepted agile maturity model exists and how the mapping of such a model to CMMI, ISO/IEC 15504 Part2 and Part 5 would be. Among 40 maturity models they include about 30 ones to the scope of the study. They group the models into those which are close to the level structure of CMMI, those which have a level structure at all and those which don't use explicit levels. In the end they emphasize the gap of a scientific research in this topic and the fact that none of the models fulfills the requirements of ISO/IEC 15504 Part2.

In 2012, Schweigert et al. conducted a survey to identify what an agile maturity model (AMM) would deliver to its users with 67 participants [77]. According to the results of the survey most of the participants think that an AMM should measure the perfect implementation of agile practices and organizational support for implementation of them. Another significant result is that more than 65% of participants think that an AMM should distinguish technical, project and organizational level processes and allow individual improvement of each process rather than a simple binary result that the organization is agile or not.

There are few studies evaluating agile maturity models. Common results of these studies that none of the current models is accepted commonly in software circles and a structured model is needed to evaluate and improve software agility.

2.4. **ISO/IEC 15504 (SPICE)**

In this subsection, we briefly describe structure of ISO/IEC 15504 and its relation to agile approach.

2.4.1. **Structure of ISO/IEC 15504**

AgilityMod's structure was defined in accordance with ISO/IEC 15504 Software Process Improvement and Capability Determination (SPICE) Model, Part 2[1] [78] and Part 5[2] [79]. Our purpose of using ISO/IEC 15504's structure is to create a common basis for performing assessments of agility and present the assessment results using a common rating scale.

In this subsection we briefly describe the structure of the ISO/IEC 15504 to create a familiarity to the reader and to better describe the similarities and differences of AgilityMod and ISO/IEC 15504.

ISO/IEC 15504 provides a structured assessment framework for software processes [80]. It facilitates process assessment, provides a basis for use in process improvement and capability determination and provides process rating which represents an objective image of current state of a process [80].

---

[1] ISO/IEC 15504 "Part 2: Performing an Assessment" provides the following copyright release:

"Users of this part of ISO/IEC 15504 may freely reproduce relevant material as part of any Process Assessment Model, or as part of any demonstration of conformance with this International Standard, so that it can be used for its intended purpose."

[2] ISO/IEC 15504 "Part 5: An exemplar Process Assessment Model" provides the following copyright release:

"Users of this part of ISO/IEC 15504 may freely reproduce the detailed descriptions contained in the exemplar assessment model as part of any tool or other material to support the performance of process assessments, so that it can be used for its intended purpose."

ISO/IEC 15504 consists of the following parts constructing an assessment framework all together:

— Part 1: Concepts and vocabulary
— Part 2: Performing an assessment
— Part 3: Guidance on performing an assessment
— Part 4: Guidance on use for process improvement and process capability determination
— Part 5: An exemplar Process Assessment Model
— Part 7: Assessment of organizational maturity

ISO/IEC 15504 Part 2 [78] which is a normative part, defines general elements for performing the assessment and describes the phases of an assessment including planning, data collection, data validation, process attribute rating, reporting and roles and responsibilities. Part 2 also describes the measurement framework for process capability (capability dimension) with all process attributes and defines the minimum rating requirements.

ISO/IEC 15504 Part 5 [79] which is an informative part, gives a detailed description of the structure of the process assessment model in conformance to the requirements defined in Part 2. Process dimension and capability dimension are described with indicators (work products and practices). Capability dimension in Part 2 is expanded to include the generic practices which are assessment indicators in Part 5.

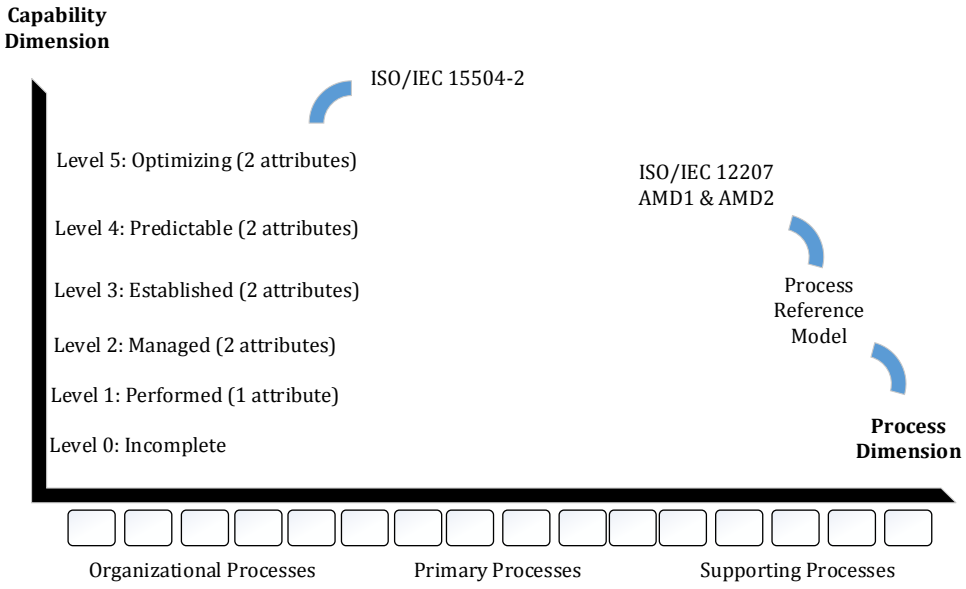

Figure 4 Process Assessment Model Structure of SPICE – adapted from [79]

ISO/IEC 15504 consists of two dimension: capability dimension and process dimension. Capability dimension defines the capability levels ranging from “Incomplete” level to “Optimizing” level as shown in Figure 4. Each level is characterized by process attributes. On the other hand, process dimension includes group of processes defined in conformance to ISO/IEC 12207 AMD1 and AMD2 [81, 82]

The process dimension includes 48 processes classified into three categories conforming the Process Reference Model ISO/IEC 12207 AMD1 [83] and AMD2 [82]. The capability dimension defines process attributes which are grouped into capability levels. There are six capability levels and nine process attributes in total as shown in Figure 4:

- Level 0: Incomplete process
- Level 1: Performed process
  - Process Attribute 1.1 Process performance
- Level 2: Managed process
  - Process Attribute 2.1 Performance management
  - Process Attribute 2.2 Work product management
- Level 3: Established process
  - Process Attribute 3.1 Process definition
  - Process Attribute 3.2 Process deployment
- Level 4: Predictable process
  - Process Attribute 4.1 Process measurement
  - Process Attribute 4.2 Process control
- Level 5: Optimizing process
  - Process Attribute 5.1 Process innovation
  - Process Attribute 5.2 Continuous optimization

The process attributes are independent of any process and applicable to all of them.

***Assessment*:** There are two types of assessment which are performed based on the assessment indicators. The first one is the process capability assessment at the capability dimension and the second one is the process performance assessment at the process dimension.

Process Capability Assessment is performed based on Process Attribute Indicators which are;

- Generic Practices (GP)
- Generic Resources (GR)
- Generic Work Products (GWP)

These indicators are valid for from level 1 to level 5.

On the other hand, Process Performance Assessment is performed based on Process Performance Indicators which are;

- Base Practices (BP)
- Work Products (WP)

Process Performance Assessment is performed only at Performed Process Level (Level 1)

Rating: Achievement level of a process attributes is rated based on a four point ordinal scale:

- Not Achieved (0-15% achievement percentage)

— Partially Achieved (16%-50% achievement percentage)
— Largely Achieved (51%-85% achievement percentage)
— Fully Achieved (86%-100% achievement percentage)

For the achievement of a capability level, assessed process attributes must be rated as largely achieved or fully achieved. [78].

We preferred to utilize rating approach of ISO/IEC 15504-Part 2 as is in AgilityMod.

2.4.2. ISO/IEC 15504 and Agility

Although CMMI (Capability Maturity Model Integrated) which is a similar model to SPICE in terms of its purpose, has been extended to be compatible with agile practices and processes, ISO/IEC 15504 has not been yet adapted based on agile values and principles. It is very obvious that current process structure of ISO/IEC 15504 does not comply with agile processes and principles.

There are few studies in the literature evaluating the relation of SPICE and agile processes. Bianco evaluates compatibility of agile manifesto and principles and SPICE profile [84]. She maps Scrum practices into processes of SPICE and concludes that with agile development it is possible for a company to achieve capability level 3. However, she misses a point that Scrum only covers a minor portion of whole software development life cycle activities. Application of Scrum does not guarantee satisfaction of all processes of exemplar model in SPICE.

Lami and Falcini performs a study to show that SPICE can be effectively used in agile contexts [85]. They discuss the applicability of SPICE in agile contexts. They conclude that organizations practicing agile techniques should not be prevented to assess its maturity using the SPICE model.

2.4.3. Reasoning Behind the Selection of ISO/IEC 15504 as a Basis Model

Current agile maturity models evaluated above do not have sound structures. Most of the models are defined in terms of level descriptions, key characteristics and assessment questions. They are unsuccessful in defining outcomes and performance indicators like practices and work products.

Although there are different ways to describe a model, we selected to use ISO/IEC 15504 as a basis for the structure of AgilityMod. Major reason of our selecting ISO/IEC 15504 as a basis is its well-defined and commonly accepted structure described above.

The structure of ISO/IEC 15504 allows separate evaluation and improvement of processes. This property brings a significant level of flexibility to organizations. In addition, there is no need to group a numbers of processes and define the rationale behind that classification.

CHAPTER 3

SOFTWARE AGILITY ASSESSMENT REFERENCE MODEL

This chapter presents the proposed Software Agility Assessment Reference Model, AgilityMod v3.0 [86]. The model was subjected to a number of updates performed following the exploratory case study and the review of agile and process improvement experts. The model presented below is the third and the last version of the model which was updated after the review of a three experts who have knowledge and experience on agile processes and process assessment topics. Previous versions of the models are published as technical reports [87, 88].

In section 3.1 we present the structure, brief descriptions of the components and rating approach of AgilityMod. In section 3.2 we explain the development progress of the model through the feedbacks of experts, their comments and the actions that are taken to improve the model. We present the full model with its dimensions in Appendix A.

3.1. Structure of AgilityMod

The model consists of two dimensions: the aspect dimension and the agility dimension which can be seen in Figure 5. In the aspect dimension, aspects are defined as Exploration, Construction, Transition and Management which are derived from agile processes and practices. In the other dimension, agility of an aspect is described with a-four-point ordinal scale which enables the agility to be assessed at “Not Implemented”, “Ad-Hoc”, “Lean” and “Effective” levels. When an aspect progresses from the bottom level: “Not Implemented” to the top level: “Effective”, its conformance to agile values and principles increases.

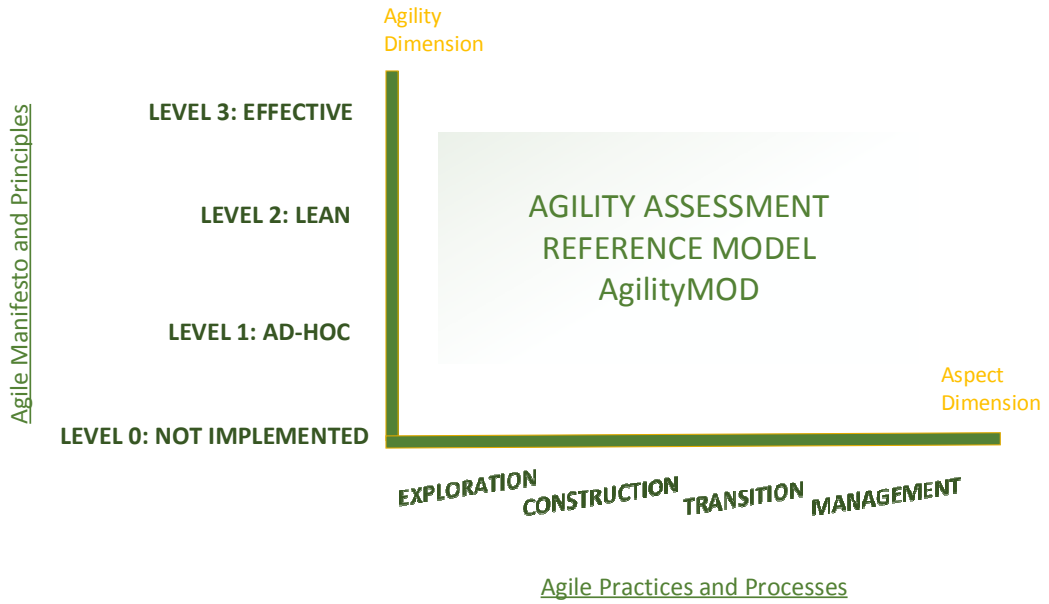

Figure 5 Dimensions of the Agility Assessment Reference Model

Assessment is performed through the aspect attributes that belongs to agility dimension of the model. Each attribute describes a major part of agility. We defined the attributes "Performing Aspect Practices" for the 1st level; "Simple" and "Iterative" for the 2nd level and "Technically Excellent" and "Learning" for the 3rd level (Figure 6).

Achievement of an agility level is assessed upon the two types of indicators: (1) agility indicators and (2) aspect performance indicators. Agility indicators indicate the application level of practices, usage level of resources and production level of generic work products. On the other hand, the single aspect attribute of the Ad-Hoc level (AA 1.1) requires the achievement of the aspect performance indicators which are aspect practices and work products. This is the only level in which the aspect performance indicators are applicable. Aspect practices and generic agility practices ensure the achievement of outcomes and outputs which are produced as a result of successfully realizing the aspect.

Agility Indicators:

— Generic Agility Practices (GAP)
— Generic Resources (GR)
— Generic Work Products (GWP/Outputs)

Aspect Performance Indicators:

— Aspect Practices (AP)
— Work Products (Outputs)

Each of these model components are described in the following section in detail.

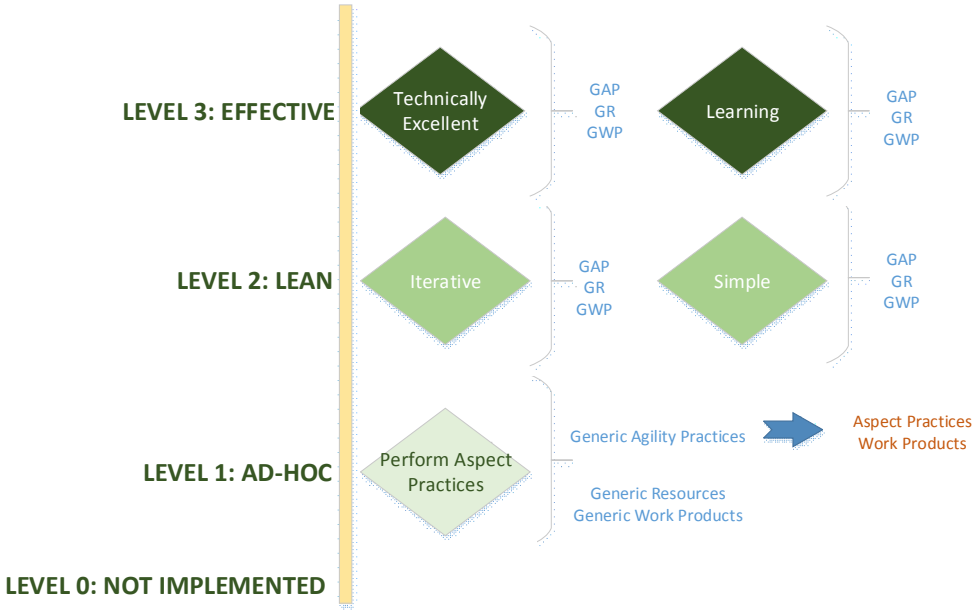

Figure 6 Aspect Attributes related to each Agility Level

3.1.1. Description of AgilityMod Components

Aspect: Formal process layers of traditional software development are intertwined to each other in agile software development. It is difficult to realize boundaries of agile processes. Aspects which are new modularization of agile processes and practices are integrated under meaningful and agile compatible abstract definitions. They are sets of interrelated and interacting activities. From this point of view, we defined four aspects fully covering a software development life cycle: Exploration, Construction, Transition, and Management.

Aspect Practice: Aspect practices are activities or activity groups that contributes to achievement of an aspect purpose and outcomes. Aspect practices also includes agile elaborations which describe how plain software development practices can be applied from an agility perspective.

Aspect Attribute: Aspect attribute is an indicator of the aspect performance. It defines the characteristic of the aspect. They are applicable to all aspect practices. Aspect attributes related to each agility level can be seen from Figure 6. All attributes are directly derived from the agile manifesto and twelve agile principles [31] by combining the related ones together. The mapping between aspect attributes and agile principles are listed below:

Table 6 Mapping of Agile Principles and Agility Attributes

No	Agile Principle	Related Aspect Attribute
[1]	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Iterative (2nd Level)

[2]	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Simple (2nd Level)
[3]	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Iterative (2nd Level)
[4]	Business people and developers must work together daily throughout the project.	Iterative and Simple (2nd Level)
[5]	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Learning (3rd Level)
[6]	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Iterative (2nd Level)
[7]	Working software is the primary measure of progress.	Iterative (2nd Level)
[8]	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Learning (3rd Level)
[9]	Continuous attention to technical excellence and good design enhances agility.	Technically Excellent (3rd Level)
[10]	Simplicity--the art of maximizing the amount of work not done--is essential.	Simple(2nd Level)
[11]	The best architectures, requirements, and designs emerge from self-organizing teams.	Learning (3rd Level)
[12]	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Learning and Technically Excellent (3rd Level)

Example Work Product: Example work products are outputs that are produced at the end of the successful achievement of an aspect or agility attribute.

Fallacy: Fallacies describe the wrong implementations which are assumed to be true.

Generic Agility Practice: Generic agility practices are activities or activity groups that contributes to achievement of an aspect attribute. Descriptions given after each generic practice specify the outcomes after a successful achievement of a practice.

Generic Resource: A kind of resource that is utilized in the conduct of an aspect or agility attribute.

Outcome: Outcomes are observable results of aspects.

3.1.2. **Mapping of AgilityMod and ISO/IEC 15504 components**

Following table displays the mapping of AgilityMod and ISO/IEC 15504 components and descriptions of the differences between the components.

Table 7 Mapping of SPICE and AgilityMod components

ISO/IEC 15504	AgilityMod	Description
Process	Aspect	Aspects inherits and include processes.
Base Practice	Aspect Practice	Base practices refer to single activities of a process, aspect practices may refer to a single activity or a process.
Process Attribute	Aspect Attribute	Attribute is a characteristic of an aspect or a process (AgilityMod uses the attribute term as is in ISO/IEC 15504)
Generic Practice	Generic Agility Practice	AgilityMod uses the generic practice term as is in ISO/IEC 15504
Generic Resource	Generic Resource	AgilityMod uses the generic resource term as is in ISO/IEC 15504
Work Product	Example Work Product	Work Products in AgilityMod differentiates from ISO/IEC 15504 as having informative characteristics rather than being normative
Purpose Statement	Purpose Statement	AgilityMod uses the purpose statement term as is in ISO/IEC 15504
Outcome	Outcome	AgilityMod uses the outcome term as is in ISO/IEC 15504

3.1.3. Agility Levels, Agility Practices and Aspect Practices

In this sub-section we provide a brief description the Model. Detailed description of all components are provided in Appendix A.

A. Agility Dimension

Agility Dimension is characterized with 4 levels: Not Implemented, Ad-Hoc, Lean and Effective. In this dimension, we describe agility levels, aspect attributes and generic agility practices, generic work products and generic resources. The figure below shows agility levels and their characteristics. For other details, please check Appendix A.

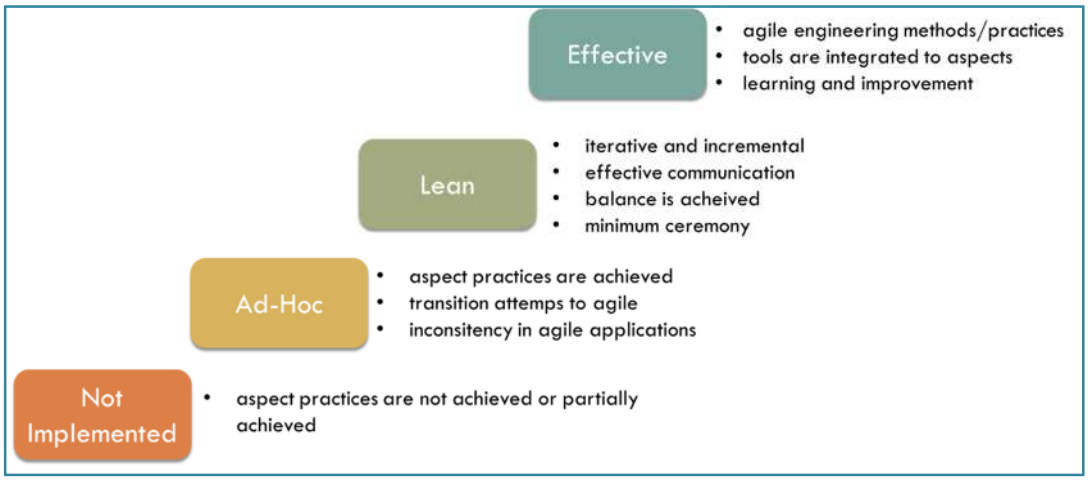

Figure 7 Agility Levels

Aspect practices either are not achieved or partially achieved at Level 0. At Level 1, organizations are capable of performing fundamental development processes such as requirements development, design, coding, integration, testing, and deployment consistently. There are transition attempts towards the agility by exploring best fitting agile practices or approaches. Aspect practices are implemented and aspect purposes are achieved; however agile values and principles are not fully incorporated into aspect practices. At Level 2 work products are developed iteratively and incrementally, non-value added activities are eliminated from the aspect practices, balance is achieved between adaptive and predictive works. At Level 3 each aspect is performed to achieve delivering value with high productivity and low defects by employing agile engineering practices and using agile tools to support a continuously improving environment.

Below we provide aspect attributes and generic agility practices related to each agility level.

Table 8 Generic Agility Practices and Aspect Attributes of Agility Levels

Agility Level	**Aspect Attribute**	**Generic Agility Practices**
Level 1: Ad-Hoc	1.1 Performing Aspect Practices	GP 1.1.1 Perform aspect practices
Level 2: Lean	2.1 Iterative	GP 2.1.1 Develop work products in an iterative and incremental way
		GP 2.1.2 Communicate effectively
	2.2 Simple	GP 2.2.1 Balance the predictive work and adaptive work
		GP 2.2.2 Employ minimally sufficient ceremony
Level 3: Effective	3.1 Technically Excellent	GP 3.1.1 Incorporate agile engineering methods/practices to the aspect practices
		GP 3.1.2 Integrate tools to aspects to improve the productivity
	3.2 Learning	GP 3.2.1 Support collaborative work and shared responsibility

		GP 3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people
		GP 3.2.3 Encourage people in the organization to participate in learning, teaching and improvement
		GP 3.2.4 Collect measures to support learning and improvement

B. Aspect Dimension

Aspect dimension is characterized with 4 aspects: Exploration, Construction, Transition and Management. In this dimension, we describe aspect purposes, outcomes, aspect practices, the relation of the aspect practices with outcomes, example work products and fallacies that needs to be avoided. Below we provide aspect practice for each aspect. For other details please check Appendix A.

Table 9 Aspect Practices based on each Aspect

Aspects	**Aspect Practices**
Exploration Aspect	E.AP1: Capture the customer and user needs
	E.AP2: Elaborate requirements artifacts
	E.AP3: Detect and resolve conflicts of requirements artifacts
	E.AP4: Specify dependencies among requirements artifacts
	E.AP5: Manage the requirement artifacts
	E.AP6: Make the artifacts visible to everyone
Construction Aspect	CN.AP1: Elaborate the work items
	CN.AP2: Explore the design
	CN.AP3: Develop the solution
	CN.AP4: Ensure the correctness of software at developer level
Transition Aspect	T.AP1: Create and Manage the Workspace
	T.AP2: Integrate the Code
	T.AP3: Deploy the solution
	T.AP4: Test the integrated solution
	T.AP5: Make the progress visible
	T.AP6: Create the supporting documentation
Management Aspect	M.AP1: Initiate the project
	M. AP2: Form the team
	M.AP3: Align and adopt the environment
	M. AP4: Establish the physical work space
	M.AP5: Plan the progress
	M.AP6: Estimate the work items
	M.AP7: Monitor the progress
	M.AP8: Manage and mitigate the risks

3.1.5. Rating Approach

In AgilityMod we are using the same rating approach defined in ISO/IEC 15504. Each practice is rated based on the following schema. For an agility level to be reached, all practices should be largely or fully achieved

- 0 : Not Achieved (Color Code: Red)
- 1: Partially Achieved (Color Code: Yellow)
- 2: Largely Achieved (Color Code: Orange)
- 3: Fully Achieved (Color Code: Green)

3.2. Development Progress of AgilityMod

In the following sub-sections we describe how AgilityMod has completed its progress of improvement and what kind of refinements are performed to the reach the final version.

3.2.1. First Version of the Model and the Exploratory Case Study Conducted

We developed the first version of the model based on an extensive research on key characteristics of agile models and agile adoption patterns. We conducted literature survey on primary agile software development models to understand the common characteristics and approach. By exploring agile models [21, 29, 58, 59, 89], we understood the agile values in practice and the reasoning behind the practices and developed *the aspect dimension* of the model. By exploring how organizations mature in agile environments [4, 32, 90-92], we developed the agility dimension of the model.

After the development of the first version of the model, we performed an exploratory case study to observe the applicability of AgilityMod in a real environment and to discover improvement opportunities related the Model. Details of the case study and the findings are described in Section 4.1. We published the case study results with a title of the "Assessing Software Agility: An Exploratory Case Study" in Software Process Improvement and Capability Determination conference (SPICE 2014) [93].

3.2.2. Second Version of the Model and Review of Experts

Based on the findings obtained in the exploratory case study we updated and published the second version of the Model [94]. We removed the "Culture" aspect from aspect dimension since its practices had in a significant conflict with "Learning" attribute practices and "Management" aspect practices and extended other attribute or aspect practices to cover the unique practices of culture aspect. At this stage, we sent AgilityMod to process improvement and agile experts to obtain their opinions about the Model and update accordingly. We received the feedbacks of three experts: Expert A: a process improvement consultant who has knowledge on agile processes from Turkey; Expert B: a SEI authorized CMMI lead appraiser who has hands on practices on agile processes and ISO/IEC 15504 from India; Expert C: a hands on agile practitioner and trainer from Canada. We asked them to review the second version of the model based on a set of criteria (fitness for purpose, completeness, definitions of agile levels, objectivity, correctness and consistency) which were defined in one of our previous studies [39]. The meaning of the each criteria which are also described in Section 2.2.1.2 are explained to reviewers.

After the review of technical report describing the Model, reviewers sent their comments through e-mails. Following that we performed a video call over Skype with the expert from India and face to face meeting with the expert from Turkey. We went through the each comment and discussed possible solutions to improve the model. These meetings are recorded for further analysis. In addition, we asked them to fill in a questionnaire to understand their overall opinion about the Model based on six criteria specified above.

Findings of Expert A:

Expert A is interested in agile software development practices and adopting kinds of approaches in software organizations more than 10 years. He has been working as an internal and external processes assessor since 2001. He has proficiency on ISO 15504, ISO 9001, ISO 27001, ISO 20000, and CMMI.

Expert A thinks that AgilityMod is a successful model bringing a maturity view on the agile principles. He mentions that using ISO/IEC 15504 as a reference model supports the validity of the model and increases the possibility of usage among organizations. He thinks that the model perfectly fits to the need of organizations that having a reference model to be utilized for getting better at agility.

We defined AgilityMod in an abstract form to embrace all agile models. Abstraction is a critical issue since high-level abstraction may cause vagueness of concepts or low-level abstraction may prevent the real purpose is not achieved. Expert A thinks that level of abstraction is at an appropriate level when the audience of the model is considered as daily agile practitioners.

He suggests the model more focus on measurement practices at higher levels of agility. In terms objectivity he thinks that the normative and informative features of the model cannot be easily understood, which makes it difficult to use as an objective resource. For example AgilityMod defines “work products” in the aspect dimension related to each aspect and it is not specified in the model if these work products are normative or informative.

He does not specify any negative comments on correctness issue and specifies minor comments on the consistency of the document.

ISO/IEC 15504-Part 7 [73] addresses the assessment of processes for organizational maturity and defines the process sets associated with each maturity level. Expert A mentions the necessity of developing a ISO/IEC 15504-Part 7 equivalent model if the Model claims to be a solution for the assessment of organizational agility.

Findings of Expert B:

Expert B has 4 years of experience on agile software development approaches but basically Scrum and more than 10 years of experience on ISO/IEC 15504. He is in software process improvement field for more than 15 years.

In his comments, Expert B questions why we selected ISO/IEC 15504 as a reference model since its process structure does not fit with agile processes. He asks how the outcomes of an aspect can be sequentially linked to with each other. He asks where

somebody should start to the improvement process. He brings to discussion to a different area that mathematical representation of the model and the relationship between an empirical process and a defined process. He specifies that he does not agree with the rating scale approach of ours, actually SPICE's.

He has comments on the order of agile levels. Based on AgilityMod an aspect becomes "lean" before being "effective". He thinks opposite of this structure and advocates being "effective" before being "lean". He explains the reason as "the whole lean concept comes in when one are already successful".

He focuses on distributed teams and the applicability of the model for distributed agile teams. He mentions that the model may need a third dimension for the handling of scaling factors like team size, geographical distribution, domain complexity etc.

He mentions the necessity that there are no gray areas in an assessment. He specifies the importance of evaluating an organization's cultural change in an agility assessment process. He makes emphasis on checking the rhythm of team and how the teams are maintaining that rhythm during the development process. In addition, he mentions that the velocity is another indicator of consistent team performance and should be checked in an assessment.

He suggests the usage of a general agile terminology rather than Scrum or XP.

He specifies that a model should include a level of tailoring, however AgilityMod's aspect dimension describes agile elaborations for activities.

In the overall, expert B evaluates the model from very different perspectives than the other reviewers. Expert A and C evaluates the model within the limits of the model. However expert B moves discussion beyond the boundaries of the model. This brings lots of critics some of which are applicable, some of which are out of the scope of this study and some of which are not relevant to the study.

Findings of Expert C

Expert C was certified as a Scrum Master in April 2006. She works since then as a consultant implementing Agile practices in various types of organizations. She also evaluates organizations' readiness to become Agile and their agile implementation for improvement. She is also co-author of a book in French on Agile. The book was awarded "Best French Informatics Book of 2012 by the Association Française d'Ingénierie de Systèmes d'Information (AFISI), whose members are voting the best book annually for over 20 years. For the last 2 years, she has been teaching agile development processes and project management as a professor at Université du Québec à Montréal.

Expert C thinks that for someone with extensive experience in process assessment, the model makes sense. However, it does not come with a related assessment method. The evaluator will need to define its own to apply the model.

She thinks that the agile principles are well covered in the model. She mentions that, it is very important to keep the model as generic as possible because not all Agile teams adopt Scrum, or Kanban, or UML (or any other method specific artifact or practice). In such

cases, she suggests that the model should be kept generic by putting emphasis on the outcomes or objective or principle.

She thinks that it is unclear how an evaluator will rate each practice to obtain a percentage value. From a measurement perspective, precisions should be brought forward in that matter to improve repeatability of the model. One way of solving that is to provide results as a range to the assessment client.

She also thinks that it would be easier to understand the model if a 3-D graphical representation is provided, showing the aspects and the dimensions. She thinks that the 2-D graphics are revealing the complexity of the model but not clarifying its complete structure.

In addition to these comments, she also specifies some inconsistencies and complicated phrases directly on the technical report.

CHAPTER 4

APPLICATION OF AGILITYMOD

This chapter presents the application of first and third versions of AgilityMod in case study settings for achieving validity. Section 4.1 describes the exploratory case study that was conducted with the first version of the model. Section 4.2 describes the design multiple case study that is going to be conducted with the third version of the model to validate AgilityMod.

4.1. Exploratory Case Study

We aimed to conduct a single exploratory case study on the first version of AgilityMod [95] prior to the review of the Model by experts in agile and process assessment field. Exploratory characteristic of the case study will not only enables us to answer the research questions given below, but also it will provide flexibility during the conduct of the case study.

Two of the objectives of this study are to investigate the applicability of the AgilityMod in assessing the aspect's agility at different levels in an organization and identifying if the Software Agility Assessment Reference Model could be used as a roadmap for organizations to improve aspects' agility. Another objective is to reveal improvement opportunities related to the AgilityMod.

Considering these objectives we identified the following research questions (RQ):

RQ1: How suitable the "Agility Assessment Reference Model_v1.0" to be used with the purpose of identifying aspects' agility?

RQ2: What are the improvement opportunities for the first version of AgilityMod?,

In the following sections we describe the design, conduct and findings of the case study.

4.1.1. Design of the Exploratory Case Study

Case Selection Strategy: Our strategy is to select the same organization that has been subject to one of our previous studies where we assessed the strengths and weaknesses of five agile maturity model/framework from agile process assessment perspective [39]. The reason of this selection is that we already had an idea about the agile maturity of the organization and knew the specific problems. From these perspectives, the organization

will enable us better to observe if the model capable of revealing these problems and indicating the agility level of the organization.

Data Collection Strategy: In the selected organization, we aim to perform gap analyzes through interviews and evidence collection and review. Interviews are planned to be performed with people from different roles/positions in accordance with five aspects of the model. These roles are planned to include at least one product owner, one business analyst, one developer, one configuration manager and one tester. We planned to record the interviews for further analyzes. We also planned to perform gap analyzes with two projects since the processes performed in the projects may differentiate, and the generalization of the assessment results through the organization is possible.

Validation Strategy: After the gap analyzes, we planned to prepare an assessment report and discuss it with the interviewees to obtain their opinion on the assessment results.

4.1.2. **Conduct of the Exploratory Case Study**

We performed the agility assessment in a government organization which is developing various management information systems related with the digitization of the procurement procedures for government purchases and health management and law tracking systems. It is a small sized company with sixty employees. The organization had been formerly assessed with other agile maturity models in 2013 by us in [39].

Prior to the assessment, we prepared the assessment questions for each aspect practices and generic agility practices.

We performed interviews with the product owner, the architect who has been formerly software development team leader and developer, the business analyst team leader and the test manager separately. Interviews took 13 person-hours in total.

During the assessment, we observed that people tend to describe positive sides of their job and positive practices; therefore, contradictory questions or failure scenarios also need to be asked related to practices. Direct evidences were also collected and reviewed in the scope of the assessment.

We used a four-level scale to express the achievement of the aspect attributes: “not achieved (0), partially achieved (1), largely achieved (2) and fully achieved (3) and not applicable (NA)”

4.1.3. **Findings of the Exploratory Case Study**

Findings are discussed in two sub-sections: findings related to agility of the organization and findings related to the AgilityMod

4.1.3.1. Findings Related to Agility of the Organization

We assessed the agility level of the organization over two types of projects: one maintenance and one new development. Figure 8 gives the colored schema of the assessment rating to capture the results at a glance.

	1. AD-HOC						2. LEAN					3. EFFECTIVE					
							Iterative		Simple			Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 2.2.3	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	2	2	2	2	2	2	1	2	0	2	1	1	2	1	2	3	2
CONSTRUCTION	2	2	2	1	NA	NA	1	2	1	2	2	0	2	1	2	1	1
TRANSITION	2	2	2	2	2	NA	1	2	1	2	3	1	1	1	2	1	1
CULTURE	2	1	0	2	3	1	0	2	XX	XX	1	XX	XX	1	1	1	1
MANAGEMENT	2	1	1	1	1	NA	1	2	0	0	1	1	0	1	2	1	1

Figure 8 Colored schema for the assessment ratings based on each practice

The numbers and the colors in each cell display the ratings given: “0” and “red” means that the practice is not achieved. “1” and “yellow” means that the practice is partially achieved. “2” and “orange” means that the practice is largely achieved. “3” and “green” means that the practice is fully achieved. For the achievement of an agility level, assessed attributes must be rated as either largely achieved or fully achieved. The result of the assessment indicates that the exploration aspect is at ad-hoc level, the construction aspect is at not implemented level, the transition aspect is at ad-hoc level, the management aspect is at “not implemented” level, and the culture aspect is at “not implemented” level.

Figure 9 displays the comparison between the current situation of the organization (inner pentagon) and the ideal case (outer pentagon) in the form of a radar chart. The data to draw the radar chart is obtained by adding the rating values given on Figure 8 for each aspect.

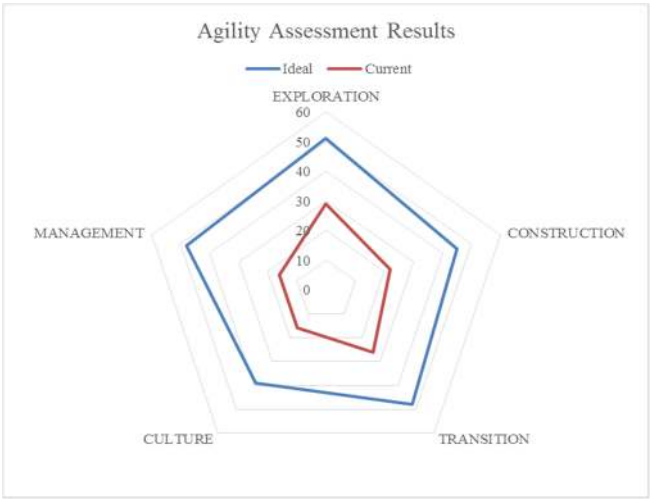

Figure 9 Comparison of the Current Situation of the Organization and Ideal Situation

Below, we briefly present the major findings for each aspect. Each aspect is questioned based on both the aspect practices belong to the level 1 and the generic agile practices belong to the level 2 and 3.

Exploration Aspect: Major findings and improvement areas identified related to exploration aspect are as follows:

- Customers do not regularly involve in the exploration activities which limits the feedback obtained at an early phase.

- There is no consistency in transforming user needs into simple requirements artifacts (user requirements, business process models, detailed use case descriptions are developed all together or not at all)
- Formal review and approval procedures are applied for scope and requirements documents which may even take the same amount of time of the analysis process.
- Detailed specifications are preventing the development team to wait for a long time before the development activities start. On the other hand, in most cases test team need to explore and learn what is going to be tested since there is nothing exist in written form after the development is completed.
- All employees work in the same open-office which enhances communication. However, lack of continuous and regular communication channels cause cases where a developer made a change on a requirement item without notification to test team or analysts until the last moment.
- The organization maintains two large systems interacting with each other. Even if the internal and external dependencies partially known, there is a significant need to identify all dependencies since the numbers of hotfixes continue growing because of the lack of knowledge on change impact.
- This is an organization where requests of customers continuously flowing to analysts or technical leads. All the requests obtained from the customers at different sizes are entered to Team Foundation Server (TFS) tool. TFS is used maintaining the requests obtained from customers at different sizes such as a backlog. However, backlog items are not differentiated based on their types and regularly groomed by the product owner.
- Collaborative working changes from the team to team and team members do not share the full responsibility.

Construction Aspect: Major findings and improvement areas identified related to construction aspect are as follows:

- Developers elaborate the items with business analysts and/or customer one by one. The testers are not involved in these activities and remain unaware of the items and solution approach especially for small and medium sized requests. If the developer does not provide enough information about the item, it may be released without testing.
- Architectural elaboration meetings are performed weekly among architects and technical leaders. However, alternative solutions are not evaluated consistently, and there are cases where redevelopment occurred.
- Static code analyzes are performed if the capacity of the development environment is sufficient. Code reviews are not performed except for the changes on critical modules.
- Physical configuration audits on source code are not performed. Therefore, it is not an unusual thing to deliver items without notice of test team or technical leads.
- Developers do not write unit tests except for mobile applications, even though, unit tests are the backbone for fast feedback.
- Experiences of the people in certain cases remain as tacit knowledge, there is no such a knowledge platform for sharing experiences in the organization.

Transition Aspect: Major findings and improvement areas identified related to transition aspect are as follows:

- Source code may be waiting for a long time in developers' branch before check-in to the mainline. Build time also varies based on the length of the code that is expected to be short in agile environments. The systems are lack of continuous integration.
- The systems are not supported with automated regression test suite. Only 6% of the whole system can be automatically tested (start-up tests).
- Functional black-box tests and regression tests are performed manually.
- It approximately takes 2-2.5 months to deploy the solution to the real system. After deployment, automated start-up tests are run. However, unknown errors may be sent to real system which is not in the scope of start-up tests causes high numbers of hot-fixes.
- Status of the integration and deployment processes are followed through e-mails. However, monitoring the progress through dashboards and making the progress visible to everyone is a better option.
- Even if the teams agree that necessary documentation needs to be produced to maintain the software, current documentation types do not help them in maintaining software since they are not continuously updated. It is obvious that the organization needs to rethink how efficiently document the product.

Management Aspect: Major findings and improvement areas identified related to management aspect are as follows:

- Before the initiation of the projects feasibility studies are performed. Scope is defined for new development type projects. A formal approval procedure and long waiting times are also valid for management type documents. Our suggestion is to use simpler forms such as one page "project data sheets" for scope and vision.
- There is not an accepted and applied estimation approach in maintenance type projects. Bottom-up estimation technique was applied in a few new development projects where every team member is involved. Estimation techniques should be reviewed considering the overall approach.
- Tracking is truly based on communication among team members and technical leaders. Such major metrics to identify team velocity is not gathered since there is no planned and actual effort.
- High-level strategic plans exist for maintenance type projects. However, there is no high level or level plans developed dynamically. Tasks are assigned people in a just-in-time fashion. Teams dynamically adapt the conditions by quickly modifying resources. This practice is useful in rush times however idle resources cannot be identified.

Culture Aspect: *Major findings and improvement areas identified related to culture aspect are as follows:*

- One of the major problems of the organization is that there is no common "Agile" perception and understanding among the people. Some believe that they are developing according to agile principles, some do not have an idea about the agile concepts.
- Most of the customers are also unaware of agile software methodologies and how and when they are going to involve the processes.

- Even though the roles are not specialized according to agile principles, teams are constructed based on a matrix structure with people from different areas and different experiences.
- Domain trainings are given by business analysts to other team members. Process trainings and technological trainings are also provided by third party consultants. There is a need to focus on agile process and practice trainings.
- Teams and projects are managed by technical leaders and project managers respectively. Resource management and leveling are performed by technical leaders whereas project managers work as administrators mostly. They are seen as non-value adding people to processes. Tasks are also assigned by technical leaders. Therefore, we cannot talk about self-organizing teams.
- Teams care about working collaboratively, however, there are examples where redevelopment is needed since the opinion of the experienced developers were not asked. Personal issues may also go upfront of the business.

4.1.3.2. Findings Related to the Agility Assessment Reference Model

We met the other purpose of this exploratory case study and identified the improvement opportunities and problems related to the first version of AgilityMod.

First of all, we observed that the model can be used for assessing organizations' agility level. Improvement suggestions given based on the model can be utilized as a roadmap for improving organizations' agility. It is also noticeable that the assessment based on AgilityMod can be performed with a reasonable effort.

We identified three types of internal problems: redundancies, missing practices, and excess of practices.

Redundancy: Aspect practices belong to the culture aspect and generic agility practices belong to 2nd and 3rd level of agility level questions the similar practices and principles. When we performed an assessment based on the generic agility practices, there is no additional need to assess the organization with practices of culture aspect.

While we design the Model's structure, we aimed that the generic agility practices will be applicable to and valid for all aspect practices. However, it seems that the practices of culture aspect and 2nd and 3rd level generic agility practices do not comply in this sense (marked with XXs in Figure 8.).

Missing practice: We observed that we do not question if the applied practices such as elicitation is project specific or applied at organization-wide in the Model.

Excess of practices: Practices of learning attribute, GP 3.2.3 and 3.2.4, are very close to each other in terms of their meaning and can be combined for a better structure.

One of the defined practices of learning attribute is "Obtain frequent feedback" which belong to 3rd level, however it more conforms to 2nd level since the purpose of 2nd level is providing feedback about progress.

We updated the AgilityMod in the light of these findings and published the model's second version to be sent to experts for review [94].

4.1.4. Validity Threats

We present the first version of AgilityMod in this study. But before the conduct of the case study we did not validate the first version of the model. Since the reason we are performing case studies is to validate the model step by step.

We designed this study as a single exploratory case study. Assessing one organization limited us to observe the applicability of the Model for different levels of agility apart from "not implemented" and "ad-hoc".

After the findings about the agility of the organization are discovered, we discussed the findings with the people who are involved in the assessment process. They are all agreed on the findings which indicated that the model can specify the problems and improvement areas of the aspects in an agile perspective.

4.2. **Multiple Case Study**

Following the update of AgilityMod based on feedback of three experts, we aimed to conduct a multiple case study to validate the model. As explained previously, we preferred to perform a staged refinement procedure on the model prior to the conduct of multiple case studies for validation. The objectives of this case study are to investigate if the proposed model can be utilized for agility assessment of software organizations and provide roadmaps to improve agility. Although they are the same with the ones in the exploratory case study, major motivation of conducting the exploratory one was the refinement of the model for a better structure.

We defined the following research questions in accordance with the objectives above:

RQ1: How suitable is the third version of Software Assessment Agility Reference Model to be used with the purpose of identifying aspects' agility, identifying the agility gaps and providing roadmaps for improving in agility in a software project?

RQ2: What are the strengths and weaknesses of the third version of AgilityMod?

4.2.1. Design of the Multiple Case Study

Case Selection Strategy: We plan to perform case studies at least five different organizations to increase the reliability of the study. For the selection of the organizations, we will pay attention to observability of every unit of AgilityMod. We will look for organizations which are at beginner level and expert level in conducting agile processes to observe every pattern in AgilityMod. We aim to select cases from different business domains to observe if agility patterns change based on business domains. We also plan to investigate the differences in agility patterns of the cases which start doing business in an agile context and the cases which have traditional software development backgrounds.

Data Collection Strategy: We plan to conduct formal assessments through semi-structured interviews with aspect owners, and evaluate direct evidences. People from different roles are planned to be involved in the interviews to obtain tacit knowledge directly from practitioners. These roles are planned to include at least one product

owner, one business analyst, one developer, one configuration manager and one tester. We plan to record the interviews for further analyzes.

Validation Strategy: After the gap analyzes, we plan to prepare assessment reports and discuss the findings with the interviewees to obtain their opinion on the assessment results.

4.2.2. Conduct of the Multiple Case Study

We applied the model in six different organizations instead of five, since we couldn't observe the high agility levels in enough detail with five case studies. Below, we describe the cases, selection reasons of the cases and findings of each case study. We kept the names of all organizations and projects private for confidentiality issues.

Rating: We used a four-level scale to express the achievement of the aspect attributes: "not achieved (0-red), partially achieved (1-yellow), largely achieved (2-orange) and fully achieved (3-green) and not applicable (NA)". For an agility level to be reached, all practices should be largely or fully achieved.

Assessment Validation: For validation of each case study result, we prepared a detailed report including current application of each practice in the project, improvement suggestions for the findings and rating of each practice. We presented the finding reports to the assessment teams, in some cases to whole project members. We obtained feedback of people who participated into the assessment for the following issues:

- if there is a misunderstood concept or practice presented in the report or presentation
- if the report or presentation covers all the improvement areas that you noticed about organization's agile processes
- if the findings presented to them are beneficial for getting better at agility
- if they follow the same improvement path suggested in the report and presentation
- which of the suggested practices are new to them or noticed previously
- and to what extent the presented findings and improvement opportunities in their projects overlap in percentage

We describe the feedbacks obtained for each case study in the following subsections.

Challenges we faced during the conduct of case studies

The level of subjective judgment: One of our aims while developing AgilityMod is to minimize the subjective judgment for the assessment of aspect and agility practices. With this purpose, we defined "Agile Elaborations" for aspect practices and described the exemplar outcomes that can be observed after successful achievement of generic agility practices (GAP). However, during the conduct of the 1st case study we observed that "communicate effectively" GAP in "Iterative" attribute and "align with agile values and principles" GAP in "Management" aspect, require asking well-structured questions and talking with whole team rather than an assessment group. We did not have chance to talk with the all project team members for these specific practices, however, we extended our question set to discover different ways of communication in the projects and detect

communication problems. For other practice, we resolved this problem by asking the same questions to different roles in the project to observe any difference in agile perception.

Required changes in the Model: Even if we updated the Model based on the results of the exploratory case study (4.1) in terms of resolving conflicts in definitions, we observed that "Balance the predictive work and adaptive work" generic agility practice and "Make the artifacts visible to everyone" aspect practice in Exploration aspect and "Make the progress visible" aspect practice in Transition aspect overlap. We updated the Model v3.0 by deleting visibility emphasis from the generic agility practice and updated the case reports accordingly.

Open-ended questions: We observed that asking open-ended questions is a better way of obtaining the tacit knowledge of team members. However, there is a side effect of this approach. You may not obtain the exact information or it may take longer than it is expected. In the best case, it is a good approach to start with open-ended questions (how do you….) and direct the assessed person with examples when things get complicated.

Evidences to be observed: Although we defined example work product for aspect practices, we agreed on the direct evidences to be observed after the conduct of the two case studies.

Getting the correct data about AgilityMod for validation: Our first case study design approach did not cover obtaining quantitative data from process owners at the validation phase. For the first three case studies we asked open-ended questions to understand people's perceptions related to our model. However, because of the difficulty in explaining the case results, we decided to ask about the success rates of the Model in terms of specifying the findings and improvement suggestions in the projects. Because of this change, we performed meetings with team members from the first three case study, explained the findings again and obtained the success percentages and the reasons behind their answers.

Meeting location: We performed the 3rd assessment via teleconferencing. However, less people are involved in the study compared to the plan. The reason was explained as the decrease in some people's motivation because of the teleconferencing approach. In addition, we couldn't observe any evidences because of this approach. We went to their offices for validation phase to overcome these issues. Face to face assessment is absolutely a better approach in terms of capturing required information, observing evidences and interaction.

4.2.2.1. 1st Case Study: Organization NM, Project 1

Organization NM is one of the leading media companies in Turkey with its 17 million unique visitors with its various internet platforms. It is located in Ankara, Turkey. We selected Organization NM since they are in an agile adoption process for 2 years and we wanted to observe the applicability of the model in a new agile adapter company. The organization mainly performs maintenance activities for the released products. We assessed one of the ongoing projects, Project #1, which both the new development and the maintenance activities are being performed. Project (and the product) #1 is an online video platform which has 10 million unique visitors and 212 million video views in a month.

In the scope of this case study we assessed aspects of Project #1 through interviews and direct observation of the evidences. Project #1 includes 9 software developers, 2 graphical user interface designers, 2 business intelligence analysts, 1 tester and 8 content providers. The assessment was performed by the author of this thesis in four-hour time with two project managers, a software team leader and a graphical user interface designer. Interviews took 10 person-hour in total.

The functional domain of the assessed project, #1, is classified as the "Controlling Information System" based on CHAR group method [96].

Findings of Case Study 1: Figure 10 gives the colored schema of the assessment ratings to capture the detailed results at a glance. Each column refers to the practices of AgilityMod. Colors and numbers in each cell refer to the achieved levels of these practices.

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	2	2	3	2	3	2	-	-	3	3	2	1	1	1	3	3	1	1
CONSTRUCTION	2	2	2	2	-	-	-	-	3	3	2	1	1	2	3	3	1	1
TRANSITION	2	3	3	1	2	2	-	-	3	3	1	2	1	2	3	3	1	1
MANAGEMENT	2	3	1	3	3	2	2	1	3	3	2	2	1	2	3	3	1	1

Figure 10 Rating of Each Practice of Case 1

The aspects, except for the Exploration and Construction, do not achieve the requirements of Level 1. This results indicate that the Exploration and Construction aspects are at Ad-Hoc Level whereas Transition and Management Aspects are at Not Implemented level for Project #1. Figure 11 displays the achieved level of each aspect in the bar chart view.

Figure 11 Achieved Agility Levels of Aspects for Case 1

Below we specify findings related to each aspect. A detailed report of this agility assessment which includes the current practices in the organization and the improvement suggestions related to each practice are provided in Appendix A.

Exploration Aspect-Case 1: Exploration aspect is at Level 1- Ad-Hoc. Positive evidences, major findings and improvement suggestions identified related to each level of exploration aspect are as follows:

Exploration 1st Level

The sources of requirements in Project #1 are the advertisement team, content providers, business intelligence team and end users of the system. Project team obtain knowledge about new features of the system through regular meetings conducted with these sources of requirements. Current running system is regularly monitored or surveys are conducted to understand the behavior and demands of end users. The features are recorded under meeting records.

- There is a need record "request" type items to a list or a tool where a unique number is assigned each of the items to follow up them later.
- Requests are elaborated with request owners. The elaborated requests are needed to be transformed into a form of requirement and are needed to be included in the requirement list.
- Dependency of a new or changed requirement to other requirements are defined based on the personal experiences of team members.
- Request and requirements are regularly prioritized. The prioritized list of requirements are needed to be made visible to whole team

Exploration 2nd Level:

Requests are obtained and elaborated in an incremental and iterative way. The cycle time for each iteration are 7 to 10 days. Iterations are regular and consistent. Communication interfaces are established between internal, external stakeholders. A real Kanban Board is utilized for all team members come together and communicate on daily activities in front of it. Informal procedures are applied for the approval of requirements. Flow of the requirements are balanced.

- A consistency in definition of the requirements are needed to be achieved. User story or use case forms can be preferred.
- Criteria to write requirement documents are needed to be identified and agreed by the whole team.
- Dependencies among requirements items are needed to be defined in a relational matrix or on the system in which the requirements are stored. This process needs to be depersonalized.
- For the identification of non-value added activities in the processes regular retrospective evaluations are needed to be performed.

Exploration 3rd Level:

Requirements are analyzed and elaborated in a collaborative environment. Team shares the whole responsibility.

When a problem or error occurred in the conduct of an aspect, the team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error.

- There is need to keep all the requests (epics), requirements (user stories), defects, technical spike needs in the form of a backlog. The backlog should describe "what" the system does rather than "how".
- Backlog are needed to be groomed regularly.
- Requirements needs to satisfy INVEST criteria. (Independent, Value-Added, Small, Estimable, Testable)
- Tools such as "Jira, VersionOne, MSF for Agile" are needed to be included in the conduct of the exploration aspect to gain improvement in efficiency.
- In the organization, people learn and improve by their own efforts. So far, they are not trained about agile approaches or practices. It is very beneficial to have an organizational training strategy and plan. Internal and external trainings are needed to be planned to facilitate agile adoption process.
- Knowledge obtained through experiences (lessons learned) are not evaluated in the project and the organization. A system to store lessons learned on requirement activities are needed to be established. Experiences are needed to be shared to whole organization.
- A comprehensive infrastructure had been established to track data real time from the running system such as number of videos viewed, video upload time etc. However, exploration aspect's activities are not followed through any measures. A measurement infrastructure are needed to be established to monitor the progress and identify improvement areas related to the exploration aspect.

Construction Aspect-Case 1: Construction aspect is at Level 1-Ad-Hoc. Positive evidences, major findings and improvement suggestions identified related to each level of construction aspect are as follows:

1st Level:

Details about high level requirements are revealed by developers. Mock-up screens are developed. Requirement sources are also involved in these elaboration studies.

- When the details are obtained about requirements, backlog are needed to be updated.

In terms of exploration of design, alternative designs are evaluated, functional dependencies among design elements are considered.

- Alternative designs needs to be evaluated not just for the graphical user interfaces, but for the architecture as well.
- Team needs to ensure that to be developed solution (design) should meet functional and non-functional requirements.
- Comments are added to code to specify changes, but not consistently. Code needs to be commented regularly considering the maintenance of the system.
- Correctness of software at developer level is ensured through manual tests performed over GUIs or consoles.
- Core review is performed but occasionally.
- The impact of a new or changed requirement on design elements are evaluated based on personal experiences.

2nd Level:

Coding activities are performed iteratively and incrementally. Communication interfaces are established between team members and other stakeholders. Team members come together discuss daily activities through daily meetings.

- Coding standards are needed to be defined and applied. The best and easiest way to ensure application of coding standards is to add static code analysis checks to process.
- Dependencies between design elements and impact of new requirements on current design elements are needed to be identified in a structured way. (i.e. Dependency Structure Matrixes)

3rd Level:

- Unit test are needed to be written in terms of developer tests to obtain rapid feedback and to construct the background of other agile practices such as refactoring and continuous integration.
- It is suggested to apply test-driven development approach to construct a more reliable system.
- Shortcut solutions to solve emergent problems create burden on code and cause technical dept. To avoid technical dept code is needed to be regularly refactored.

Collaboration among team members are seen in development activities as well. Developers select their own tasks to work. Development team leader leads the team in technical issues.

- As a part of collaborative work, it is suggested that every team member could be involved in testing activities at code level.
- A knowledge management system should be established to share learned knowledge and organizational memory covering coding activities.
- Agile metrics are needed to be defined and collected for construction activities.

Transition Aspect-Case 1: Transition aspect is at Level 0-Not Implemented. Positive evidences, major findings and improvement suggestions identified related to each level of transition aspect are as follows:

1st Level:

In project #1, code is under configuration control. There are two environments for software development: "development" and "production" environments. However, for coding and testing activities the same environment "development" is being used. Check-in, check-out mechanisms of GIT tool are being used for labeling and versioning of the code.

- Uncontrolled change is done in the code. A system should be established in order to link requirements or tasks to changesets (code units) or unique numbers of the requirements or tasks should be added to the code as comments.

Code sets that are being developed parallel in developers' local computers, integrated in the "development" environment. (At this we evaluate only parallel computing and

correct integration). Code is automatically deployed to the "production" environment that is a facility of PHP language.

- Test engineer tests the application through manual tests in the "development" environment where coding continuous in parallel. This unstable environment cause the test be unreliable.
- Found bugs are not stored or retested: Bugs should be stored and retested by tester or to eliminate manual testing and increase test coverage, unit tests should be written immediately where a bug is found.
- Transition activities especially integration status and deployment status are needed to be made visible to whole team to provide feedback, improve transparency and collaboration.
- Supporting documentation should be developed for the maintenance of the system or essential information should recorded on the software tool where information is available when needed.

2nd Level:

Transition aspect activities are performed iteratively and incrementally. Team members share the same room and effectively communicate.

- To balance the predictive work and adaptive work, test preparation activities should be performed. Test cases are needed to be specified. (It can be started with regression test cases)
- Another suggestion to achieve the balance is to start development activities by coding tests.
- For the identification of non-value added activities in transition aspect, regular retrospective studies are needed to be performed.

3rd Level:

- Using the same environment for coding and testing activities where only manual GUI tests are performed for testing may cause serious issues in quality of software. To solve this problem either these environments should be separated or as better option test driven development (TDD) are preferred.
- After the system are strengthen with unit tests, continuous integration (frequent commits to the mainline. Compilation of all code and running of all automated tests with every check-out of the code unit) should be applied to gain rapid feedback capability and to keep the mainline always in a working state.
- There is no automated tests running at the background. An automated test suite is needed to be developed.
- An agile metric system is needed to be defined for the monitoring of transition activities.

Management Aspect-Case 1: Management aspect is at Level 0-Not Implemented. Positive evidences, major findings and improvement suggestions identified related to each level of management aspect are as follows:

1st Level:

- Project vision is defined as "to provide a video experience to the user". The vision is needed to be revised with technical improvements.
- In terms of agile awareness, internal and external stakeholders are familiar to the idea of agile software development. However, there is no consensus in the organization yet. Both internal and external stakeholders are needed to be informed about the agile values, a consensus should be established.
- Whole team works in the same open office. Physical conditions of the office provides an open communication environment.
- Project managers develop high level plans which are elaborated and detailed with every iteration. Daily activities are coordinated with the team through daily standup meetings in front of the Kanban Board.
- Risks are not tracked regularly. Examples show that corrective actions are taken after the occurrence of the risks.

2nd Level:

Management aspect activities are performed iteratively and incrementally. Team members share the same room and effectively communicate. Decision are taken collaboratively, team shares the responsibility. The balance is achieved for the management activities. For the identification of non-value added activities, regular retrospective studies are needed to be performed.

3rd Level:

- For features or requirements effort is estimated based on expert estimation. Size needs to be estimated besides effort estimation. An estimation approach is needed to be established.
- The information kept in real Kanban board can be transferred to management tools that provide effective management and tracking over metrics.
- Agile metrics are needed to be defined and collected for management activities.
- A knowledge management system should be established to share learned knowledge and organizational memory covering management activities.

4.2.2.2. 2nd Case Study: Organization G, Project #2

Organization G is a government IT organization responsible from developing e-government software to various governance organizations, located in Ankara, Turkey. Project #2 is also an e-government project providing solutions to a ministry department, 40 foundations located in different cities of Turkey and approximately 25 million Turkish citizens.

To briefly give information about the project; Project #2 includes 21 currently working employees divided into four teams which report to a project manager and an assistant project manager. Three of these teams purely works on development of software modules, the last one both deals with system infrastructure and development of the activities. Each team includes a technical team leader. Other members of the team does not have specific roles, each one is involved in analysis, design and development activities. Since the beginning of the project, 2009, 7 million LOC has been developed. Each iteration is 1 month-length. There is a signed contract between the organization and the customer specifying the dates and budget.

In the scope of this case study we assessed aspects of Project #2 through interviews and direct observation of the evidences. The assessment was performed in three-hour time with the technical leader of the infrastructure team who had worked as developer formerly and has knowledge about project's processes. Interviews took 6 person-hours in total.

The functional domain of the assessed project, is classified as the "Controlling Data System" based on CHAR group method [96].

Findings of Case Study 2: Figure 12 gives the colored schema of the assessment ratings to capture the detailed results at a glance. Each column refers to the practices of AgilityMod. Colors and numbers in each cell refer to the achieved levels of these practices.

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	3	3	3	3	3	3	-	-	3	2	2	2	3	3	3	3	2	2
CONSTRUCTION	3	3	3	3	-	-	-	-	3	3	3	2	3	3	3	3	2	3
TRANSITION	3	3	3	3	3	3	-	-	3	2	3	3	3	3	3	3	2	3
MANAGEMENT	3	3	2	2	3	3	2	2	3	3	3	2	3	3	3	3	2	3

Figure 12 Rating of Each Practice of Case 2

The assessment provides very promising results for Project #2. All of the aspects meets the requirements of Level-3: Effective. To simply specify, Project #2's aspects are iteratively performed, lean, technically excellent and continuously improving. Fast feedback is obtained and effectively communicated among team members. Figure 13 displays the achieved level of each aspect in the bar chart view.

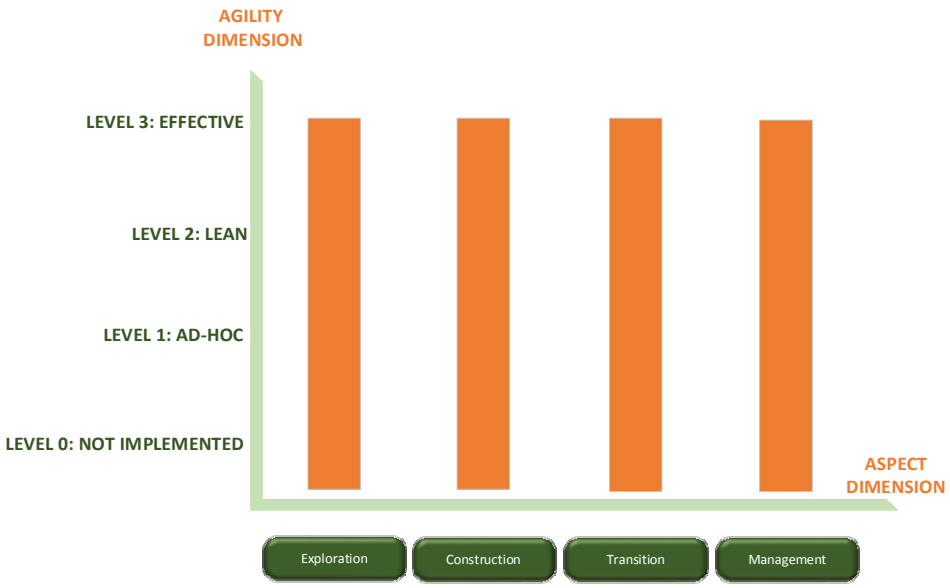

Figure 13 Achieved Agility Levels of Aspects for Case 2

We specify the assessment findings below for each aspect and each agility level. The full assessment report is provided in [97].

Exploration Aspect-Case 2: The appraisal results showed that the Exploration aspect is at Level 3- Effective. Among sixteen practices, eleven practices are rated as fully achieved and five practices are largely achieved. Below we present the positive evidences and improvement suggestions for the practices that are largely achieved.

Exploration-1st Level:

Capturing customer and business needs: In Project #2 requirements of the system are obtained from the specialists from ministry and end users of the system. Each development team are responsible for capturing the business needs for related modules. Project teams and customers are located in different buildings in the same city. Technical lead and team members obtain tacit knowledge of the customer through meetings in customer environment. End users are also invited to these meetings when required. Following that teams develop scenarios and take the approval of the customer on these scenarios for a specific iteration. The size of the requirements obtained from the customer may not be just for one month long.

These activities meets the requirements of first level for capturing customer and business needs practice.

Elaborating requirements artifacts: Requirements are elaborated within the teams based on the information obtained from customers. User stories are being developed. Team does not wait until all the requirements are being developed for the iteration, instead completed requirements move to the production line immediately. Non-functional requirements are also transformed into user stories or scenarios.

Detecting and resolving conflicts of requirements artifacts: The conflicts on requirements are resolved through communication. Customer is available through e-mail or phone in daytime. Additional meetings with customer may be requested to resolve issues when necessary.

Specifying dependencies among requirements artifacts: Dependencies among requirement items and business rules are established in Atlassian's Confluence tool.

Managing the requirement artifacts: A backlog is maintained, updated and estimated at the beginning of each iteration. Each team has its own backlog. Issue type items are maintained in Atlassian's Jira tool to specify the development of specific requirements.

Managing the change in requirements: The change information is maintained at Jira tool over "issue" items. If the changed issue is at "close" state when the change action is emerged, a new issue is opened at Jira. If not, the current issue is updated accordingly. All team members are informed about the change of the issue through communication.

Prioritization: Backlog is prioritized based on the value of the issue. Most valuable items are developed in the first order.

Making requirements artifacts visible to everyone: Jira's dashboard view is being used to capture the situation of the ongoing requirements. The dashboard includes "backlog, new, in progress, done and verified" columns filled with issues in prioritized order. The picture of each person assigned to the items can be viewed on the item.

Exploration 2nd Level:

Developing work products in an iterative and incremental way: Exploration aspect activities are performed in an iterative and incremental way. Each iteration is 1 month-long.

Communicating effectively: Communication interfaces are established between internal, external stakeholders. All team members come together and communicate on daily activities through daily stand-up meetings. 4 teams work in 4 different rooms, however rooms are connected with windows instead of walls to improve communication. Since the customer are located in a different building, communication is not effective as the communication among team members. The ways to communicate with customer are needed to be elaborated.

*Balancing the predictive work and adaptive work***:** Teams position themselves in terms of deadlines based on the dates on the contract. When the date for a specific module arrives team contacts with the customer and arranges new meetings to obtain requirements. However, there are also situations where customer do not provide quick feedback. Development team may wait for the new requirements and approval of the requirements. Therefore the role of the customer and the impact of their late feedback in development process are needed to be described in a better way to the customer. A better collaboration are needed to be established with the customer.

Employing minimally sufficient ceremony: Formal approve mechanisms between customer and project team may slow down the moving to the development. Heavyweight, not value added practices are eliminated from the processes. An external quality assurance team assesses the processes of the project regularly. Processes are also being evaluated and improvement actions are being taken. For example project teams decided to write user stories instead of use cases. It is suggested to performed regular retrospective studies that would provide much more value to the processes.

Exploration 3rd Level:

Incorporating agile engineering methods/practices to the aspect practices: Applications and practices in Project #2 covers the needs of technical excellence attribute. A backlog is being used and groomed regularly. Requirements are written in the form of user stories. Properness of the requirements to INVEST criteria (Independent, Value-Added, Small, Estimable, Testable) are needed to be evaluated.

Integrating tools to aspects to improve the productivity: In Project #2, tools are effectively used to improve the management of requirements artifacts and improve the productivity. "Confluence" tool is used to store requirements. Additional documents for requirements are not kept. "Jira" tool is used to keep backlog and track items.

Support collaborative work and shared responsibility: All team members are responsible from exploration activities. There is collaboration and shared responsibility in the team. Issues are submitted to "issue pool", no task assignment made to team members except for the issues which require specialty. However, team leaders whose approaches may differentiate may chose different types of management approaches, pooling mechanism is one the approaches being used successfully.

Adopt agile leadership styles and adjust the behaviors towards mistakes of people: Project managers delegates the job to development teams and not involved in technical details or assignment. They obtain feedback from technical leaders of each development team. They don't dictate any process or task to development teams. They coordinate the activities with upper-level managers.

When a problem or error occurred in the process, team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error. No-one is blamed.

Encourage people in the organization to participate in learning, teaching and improvement: An organizational training strategy is available in the organization. Based on these strategy every employee is trained based on his/her work area. Teams keep lessons learned in Confluence tool, however not effectively maintained. Lessons learned are needed to be updated regularly at the end of each iteration.

Collect measures to support learning and improvement: Metrics are regularly collected analyzed for exploration aspect activities. Story points related to each user story are kept and monitored in one of the development teams. Other development teams are needed to adapt measurement approach of the agile development team.

Construction Aspect-Case 2: The appraisal results showed that the Construction aspect is at Level 3- Effective. Among fourteen practices, twelve practices are rated as fully achieved and two practices are largely achieved. Below we present the positive evidences obtained from the case study and improvement suggestions for the practices that are largely achieved.

Construction Aspect - 1st Level:

For *the elaboration of work items*, team members obtain detailed information before building a solution either discussing with customer within the team. Mock-up screens are developed.

In terms of design exploration, alternative solutions are evaluated before the development of the architecture at the beginning of the project. If the detailed design of the solution is not obvious, team draws the design (class and sequence diagrams) in Confluence tool and discuss on it. Technical search activities are performed at a high component level. The impact of new arriving requirements on modules and a lower level module components are evaluated based on personal experiences.

J2EE is utilized for coding the server side and Felix is utilized for coding the client side. Coding standards are applied to improve the code quality. The system does not allow check-out of the code without writing a comment. However, comments are not reviewed. The efficiency of the code comments are needed to be evaluated. This will increase the clarity of the code especially at the maintenance phase.

To *ensure the correctness of code at developer level*, developers writes unit tests before the code. Code is frequently refactored since unit tests and automated integration tests are available in the system.

Construction Aspect 2nd Level:

Coding activities are performed iteratively and incrementally.

In terms of communication effectiveness, all team members come together and communicate on daily activities through daily stand-up meetings. 4 teams work in 4 different rooms, however rooms are connected with windows instead of wall to improve communication. Communication among development teams are established over team leaders. Design decisions and code parts are discussed and reviewed over Crucible tool. Everyone can comment on the code part and all comments are seen by other reviewers. After the review, final remarks are done with a meeting.

Team balances the adaptive work and predictive work for design and development activities. Decisions are tried to be taken at last responsible moment. Prototyping is performed to observe if the suggested solution works or not. Static code analysis to check the conformance to coding standards are performed regularly.

Employing minimally sufficient ceremony: It was observed that the way dependency and risk analysis on design elements violates the employing minimally sufficient ceremony practice at this level. Dependencies between design elements are specified in Confluence tool at higher abstraction level. Proper abstraction level for dependency analysis are needed to be specified. A matrix structure are needed to be developed to view the impact of requirements on design elements and the relations among design elements themselves. An efficient approach to evaluate dependencies among design elements should be defined and decided approach are needed to be applied the whole system. (such as Dependency Structure Matrixes)

Construction Aspect - 3rd Level:

Incorporate agile engineering practices to the aspect practices: This practice is fully achieved because of the following adaptations: Team applies the rules of coding standards. Code and design reviews are being performed through Crucible tool for major, risky or significant development activities. Refactoring issues are opened after the reviews. Test Driven Development is preferred in some cases. It is kept optional in development teams. Code is covered with unit tests.

In some cases, teams need to develop quick solutions or hot fixes which may result in technical dept. Recovering from the technical dept is the responsibility of the developer who develops it. There is no specific mechanism to control it. Our suggestion is to improve the ability to manage technical dept. Static code analysis and code reviews may be utilized to identify technical depts.

Integrate tools to aspects to improve the productivity: SVN tool and check-in, check-out mechanisms are being used for version control. “Sonar” tool is being used for static code analysis. “Crucible” tool is being used for peer code review. “Gliffy” tool are preferred for design development. “Eclipse” is used for code development.

Learning Attribute Achievements:

All team members are involved in coding and unit testing practices. Every team member has right to change the code. Collaboration among team members are seen in development activities as well. Developers select their own tasks to work except for the ones that require specialties.

Test Driven Development and Database trainings were taken by everyone in the team. People share their knowledge and learn from each other.

Code quality metrics are kept and evaluated through Sonar tool. Code coverage for each module is one of the monitored metrics.

Transition Aspect-Case 2: The appraisal results showed that the Transition aspect is at Level 3- Effective. Among sixteen practices, fourteen practices are rated as fully achieved and two practices are largely achieved. Transition aspect is the most successful aspect of Project #2. Below we present the positive evidences obtained from the evaluation of this aspect and improvement suggestions for the practices that are largely achieved.

Transition Aspect-1st Level:

The practices that are described in Level 1 also meet the expectations and requirements of Level 2 and Level 3 for transition aspect.

Creating and Managing the Workspace: Code is under configuration control. There are three branches in the development environment. These are "development (trunk)", "test" and "production". Check-in, check-out mechanisms are being used to change the code. Code is labeled and versioned with every change. Code cannot be committed without commenting. Change sets in SVN are linked to the issue items on Jira that means the reason to change code is known.

Integrating the Code: Code sets are being integrated in the "development" environment after they are developed in developers' local computers. It is a good idea to perform physical code configuration control to audit the changes in the code before the build.

Deploying the solution: At the end of each day, developed solutions are being deployed to an application server. In this application server all automated GUI tests are run all through the night. Automated reports are generated obtained for these automated GUI tests. Code packets is automatically deployed to trunk and test branches, however, it is preferred to deploy to the production environment manually to control database changes and reduce risks. Redeployment because of errors rarely occurs.

Testing the integrated solution: Automated tests are run every night. These tests are developed by team members in Rannorex tool. In addition to automated tests manual exploratory tests are performed. The bug report created by Rannorex tool, examined every day by the team leaders or a delegated person. The reason of the bug is specified and issue record is opened in Jira tool if it is required.

Transition Aspect-2nd Level:

Transition aspect activities are performed iteratively and incrementally. Team members share the same room and effectively communicate. Information radiators that are usually utilized to specify the person who commits to the mainline can be used.

Team balances the adaptive work and predictive work for transition activities by applying continuous integration strategy. There is also a continuous testing process.

Acceptance tests in which the customer is involved are performed only for new modules or new pages where the development process may take longer than a month. In these cases, teams obtain early feedback from the customer and do not wait until the module or development of new page finishes. Balancing the flow of the work is achieved.

Transition Aspect - 3rd Level:

Incorporate agile engineering practices to the aspect practices: TDD and continuous integration activities are performed consistently. Tests are run continuously. Automated regression tests suite exists. No manual tests are performed except for the exploratory tests.

Integrate tools to aspects to improve the productivity: It is ensured that codebase is working or not with the usage of Jenkins tool. Jenkins tool checks the code with every 5 seconds, if a new committed code arrives, it compiles the code, runs unit tests and integration tests, and emails the problems to team leaders related to the integration. Rannorex tool is utilized for development of automated tests. Confluence tool for information sharing.

Transition aspect's metrics are collected and monitored over Jenkins tool. For example, the number of the successful and failed builds. There is a continuous improvement in the organization. Formerly a separate test team run functional tests manually. However, it was observed that this is so demotivating for computer engineers. Then this approach was abandoned and decided everyone to involve in development and testing activities and quitting manual tests.

Here to mention that we don't share the same positive evidences and findings that intersect with previous aspects.

Management Aspect-Case 2: The appraisal results showed that the Management aspect is at Level 3- Effective. Among eighteen practices, fourteen practices are rated as fully achieved and six practices are largely achieved. Below we present the positive evidences obtained from the evaluation of this aspect and improvement suggestions for the practices that are largely achieved.

Management Aspect-1st Level:

At the beginning of the project a comprehensive feasibility study was performed. Technical challenges were specified, effort and budget were estimated at a high level. The purpose of the project was identified and project charter was developed. In addition

a contract was developed and signed with the customer. The contract has been a living document throughout the project.

While forming the team, it was one of the critical factors for Project #2 that everyone can involve coding activities. Personal specialties are taken into account while forming the teams like database specialists, configuration management specialists.

Internal stakeholders are aware of the agile values and principles. Customers are not directly informed about agile processes or principles. However, they are involved in the processes as discussed previously. The importance of the feedback obtained from the customer may be emphasized more clearly. By this way latencies may be eliminated in the approval procedures.

Establishing the physical workspace: Offices are open and sometimes the environment can be very loudly. There is not enough room to get rest and sit and drink your coffee. There is enough air and light in the environment. Physical workspace conditions can be improved. Quite workspaces can be arranged for the team members who needs privacy. A consensus may be established for being more quite.

Planning: A high level plan was established following the contract. ISBSG data set and COCOMO approach was used to estimate the effort and schedule. Effort and budget were updated based on the new information and improvements. Schedule includes the start and deployment dates of modules. Project managers transmits the information to the development team. From this moment, technical team leaders are responsible from the internal planning activities. Daily activities are coordinated through daily stand-up meetings within the teams. A bottom approach for effort and schedule estimation based on historical data of the organization would be more reliable.

Estimating the work items: Teams perform story point estimation and apply poker planning approach to estimate the job to be done within an iteration. Story points (SP) (completed SP vs remaining SP) are updated regularly through daily standup meetings.

At the beginning of the project, it is necessary to record actual effort values to Jira to identify if the team meets the estimation. However, this approach was abandoned since the estimations are more accurate. Features are re-estimated when change occurs. In order to construct a reliable historical dataset for size and effort, actual data are needed to be regularly recorded.

Progress of the teams are monitored through meetings performed daily or 3 times a week (changes based on the team). There are teams who are consistently applying the velocity of the team though Burn-down charts. Technical team leaders inform the project managers about the progress. Project monitoring meetings are performed with every 2 to 3 months with the customer.

Project risks are not tracked regularly. When a risk realized, a person is assigned to it. Risk monitoring approaches are needed to be evaluated. Risks may be tracked over Jira.

Management Aspect - 2nd Level:

Managerial tasks or documents do not prevent and cause lateness in development activities. Project managers are responsible from the development of required plans, reports or else. Balanced is achieved.

Management Aspect - 3rd Level:

Effort and size is estimated. Planning and monitoring are performed continuously. Jira, Confluence and MS Project tools are utilized for the conduct of management aspect practices. Collaboration among team members are seen in planning and estimation activities. Estimation at low level is performed by each team member. Management aspect's activities are followed through measures. Deviations between planned vs actual effort and size are two of them.

4.2.2.3. 3rd Case Study: Organization L, Project #3

Organization L is the largest independent software company in Turkey. With its various products organization L serves for 1.300.000 end users and 170.000 companies. It is located in Gebze, Turkey. In the scope of this assessment we evaluated the aspects of an ERP product developed on Windows infrastructure. Due to the confidentiality issues, the name and the scope of the assessed organization and project are kept private. We named the assessed project as "Project #3".

Project team includes 19 full-time employees. Since the beginning of the project, 2005, approximately 6 million LOC have been developed with Pascal language on Delphi platform. The iteration length of Project #3 is 7 week-long. For the development of ERP system, business needs and requirements are obtained from both internal and external customers. The project team includes one product owner, one scrum master, one business analyst, eleven developers divided into two teams and five testers. There is no project manager role in the team.

In the scope of this case study we assessed aspects of Project #3 through interviews. The assessment was performed with the scrum master, the product development manager and the product owner over skype in 4 hour-time. The assessment took 20 person-hours in total. After the assessment we specified the findings for aspects and presented the findings in a face to face meeting where six people were involved including the product development manager, test team manager, software development director of the organization, product owner, scrum master and a tester. We found change to observe a sample of direct evidences related to the aspects before the presentation.

The functional domain of the Project #3, is classified as the "Non-Specific (Complex) System" based on CHAR group method [96].

Findings of Case Study 3: Figure 14 gives the colored schema of the assessment ratings to capture the detailed results at a glance. Each column refers to the practices of AgilityMod. Colors and numbers in each cell refer to the achieved levels of these practices.

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	3	2	3	2	3	2	-	-	2	2	2	2	1	1	1	2	3	3
CONSTRUCTION	3	2	3	3	-	-	-	-	3	2	2	2	1	2	1	2	3	2
TRANSITION	3	3	3	3	2	3	-	-	2	2	1	2	1	1	1	2	2	1
MANAGEMENT	3	3	3	3	3	3	3	2	3	3	3	2	2	3	2	2	3	3

Figure 14 Rating of Each Practice of Case 3

Before we conducted the appraisal in project #3, the scrum master of project #3 had presented their agile methodology in a national conference, claiming that how good they are in agile. The results of the appraisal show that Exploration and Construction aspects are at Lean level whereas Transition aspect is at Ah-Hoc level and Management Aspects is at Effective level for Project #3. Figure 15 displays the achieved level of each aspect in the bar chart view. Their adjusted methodology of agile actually stands on Scrum method. That's way they are good at performing management aspect practices. They are frustrated about the assessment results. We can't say that they are doing agile the overall, but they are managing the project in an agile way and there are lots of improvement opportunities for them in other aspect. Figure 15 displays the achieved level of each aspect in the bar chart view.

Figure 15 Achieved Agility Levels of Aspects for Case 3

We specify the assessment findings below for each aspect and each agility level. The full assessment report is provided in [97].

Exploration Aspect-Case 3: The assessment results showed that the Exploration aspect of Case 3 is at Level 2- Lean. Among sixteen practices, five practices are rated as fully achieved, eight practices are largely achieved and three is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Exploration Aspect - 1st Level:

Capturing customer and business needs: The product developed in the scope of Project #3 is a living product. Released versions of the product are being used by the business partners and end users. Addition of new features to the product and maintenance jobs

are managed in parallel. New requests or bugs are obtained from end users and business partners through an internally developed software tool. In addition, request may be delivered via a call center and e-mail. When needed, business analysts meet with the business partners and detail "new" type of requests to understand the needs better. In addition, competitor products are evaluated to identify new development opportunities.

Elaborating requirements artifacts: The requests and requirements are elaborated by business analysts. Large-sized requests are divided into phases or modules. Elaboration of a group of requests may take a full sprint long which is a sign of a problem. Mock-up screens and detailed specifications are developed. Business analysts are responsible from the development of relational database structure. Developer use this relational database structure as a base for coding practices. Non-functional requirements are transformed into user stories or scenarios. All items are needed to be elaborated at a similar level of granularity. When the elaboration activities took a sprint long, the cycle turns to a mini-waterfall life cycle.

Detecting and resolving conflicts of requirements artifacts: Developers and business analysts communicate on requirements artifacts to resolve any conflicting issue. The requirement documents are reviewed by product development managers and product owners. When a problem is identified, documents are updated accordingly.

Dependencies among requirement items and business rules are established in Jira tool.

Managing the requirement artifacts: Defects and new development type requests are included in the backlog type list in Jira. Items are maintained in Jira as "issues". "The list" may contain items at different granularity level. The items on the list are groomed with every sprint planning meeting.

The changes in requirements are managed via Jira tool. A new issue is opened to the Jira when a change is induced and an email sent to analysis group to review the new item. All related parties are needed to be informed about the change, not just the assigned person and the analysis group. Face to face communication is the best approach to be informed in these situations. The impact of the changed artifact in terms of effort and schedule is needed to be specified.

Making requirements artifacts visible to everyone: "Issue" type items can be queried and listed based on the type of the planned release version of the product on Jira tool. However, the "product backlog" and "sprint backlog" are needed to be made visible to everyone in the prioritized order.

Even if the described scenario above for exploration aspect show problematic areas, these activities meet the requirements of first level agility.

Exploration Aspect-2nd Level:

Exploration aspect activities are performed in an iterative and incremental way. However, some requirement analysis activities may take a sprint-long and feedback are obtained after a 7 weeks analysis process. Therefore we cannot talk about a fast feedback cycle and consistency in exploration aspect activities.

In terms of communication efficiency; the team performs weekly scrum meetings to discuss the situation of the ongoing tasks. Sprint planning meetings in which the product owner, business analysts and the team leaders are involved are performed. A developer is invited to these meetings each time. Customers are informed before and after their request are developed and deployed.

Teams are formed based on functionalities and work in separate rooms. As far as we observed the communication interfaces are very limited and teams mostly communicate over documents. Therefore we suggest the elaboration of communication ways.

Balancing the predictive and adaptive work: Requirement specification documents are developed in detail, reviewed by product development managers and then coding activities start for new development type, large-sized issues. For bugs and small issues developer do not have to wait much. Instead of waiting for all requirements to complete, requirements can be moved to the product line. Detailed documents are in the trade-off between verbal and written communication.

Employing minimally sufficient ceremony in exploration aspect activities: Requirement specification documents are reviewed by product development managers and product owner from functional and technical perspectives. Review procedures may take 1-week long. Processes are evaluated and improvement actions are taken to eliminate heavyweight procedures. Formal approval mechanisms may slowdown the move to the development process and lessen agility.

Exploration Aspect-3rd Level:

Incorporating agile engineering methods/practices and tools into exploration aspect practices: Backlog type item list is maintained and groomed regularly with every sprint planning meeting. However, requirements are not defined in a consistent format. In addition, appropriateness of requirements to INVEST criteria (Independent, Value-Added, Small, Estimable, Testable) are needed to be evaluated. Utilized backlog structure in Jira needs to be elaborated. One of the Jira plug-ins can be utilized to specify requirements on Jira. Issues and requirements can be directly linked over Jira.

Supporting collaborative work and shared responsibility: The interview we conducted with product owner who is the head of business analysis team and the product development manager showed that two teams have their own concerns and not moving together towards a shared goal. Business analysts specify every detail in requirements even the database relations. It was felt that developers have a tendency to throw the responsibility to analysts when an issue occurs. These are negative evidences for the collaboration among team members.

Agile leadership: Product development managers and product owners assign the tasks to other team members. At the best scenario, it is suggested every team member select their own tasks from an issue pool rather than the assignment of tasks from managers.

When a problem or error occurred, the person responsible from the mistake is invited and asked about the problem without blaming. The purpose is to understand the reason of the problem and discover improvement suggestions.

Organizational learning: Actions are taken to improve exploration aspect activities. An organizational training strategy is available in the organization. All employees are given trainings about the foundations of agile software development and each new employee obtain the same training. All product owners took external Scrum trainings. Lessons learned are kept in a wiki-based web page and regularly updated.

Metrics related to the exploration aspect activities such as planned and completed story points are collected and monitored regularly.

Construction Aspect- Case 3: The assessment results showed that the Construction aspect of Case 3 is at Level 2- Lean. Among fourteen practices, five practices are rated as fully achieved, six practices are largely achieved and two is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Construction Aspect - 1st Level:

For *elaboration of work items*, business analysts provide detailed information before development. In *terms of design exploration*, developers perform technical search activities based on experiences. Dependencies between design elements are not specified. Modeling options at high level and detailed level design are needed to be elaborated. Alternative design (not GUI) solutions are needed to be evaluated.

Pascal language is utilized for coding standards are applied to improve the code quality. It is suggested to evaluate efficiency of the code comments for the clarity that may be required in maintenance phase of the project.

Developers tests the correctness of the code through manual graphical user interface tests.

Construction Aspect - 2nd Level:

Construction aspect activities are performed iteratively and incrementally. Weekly scrum meetings are conducted within the development team.

Team balances the adaptive work and predictive work for design and development activities. Risk evaluation of design is not performed in a formal way. Risk evaluation activities on design are needed to be conducted regularly.

Static code analysis to check the conformance to coding standards are performed regularly to support balancing the flow of the work. Dependencies between design elements, are specified based on experiences. Proper abstraction level for dependency analysis are needed to be specified. A matrix structure are needed to be developed to view the impact of requirements on design elements and the relations among design elements themselves. An efficient approach to evaluate dependencies among design elements should be defined and decided approach are needed to be applied the whole system. (i.e. Dependency Structure Matrixes)

Construction Aspect - 3rd Level:

Technical Excellence Attribute:

Incorporating agile engineering practices to the aspect practices: Team applies the rules of coding standards. Static code analysis is performed. However, code is not refactored continuously and regularly. Unit tests are not developed by developers.

Ability to manage technical dept are needed to be improved. To be able to perform refactoring continuously software should be supported with unit tests. Code and design reviews can be performed through Crucible tool for major, risky or significant development activities. Refactoring issues might be opened after the design reviews. Code coverage can be calculated after unit tests are written.

Integrating tools to construction aspect to improve the productivity: An SVN tool and check-in, check-out mechanisms are being used for version control of code. "Sonar" tool is being used for static code analysis. "Crucible" tool can be used for peer code review. "Gliffy" tool can be preferred for design development.

Learning Attribute:

Development team is responsible for coding activities. It mentioned that there is collaboration among development team members. Code quality metrics are kept and evaluated through Sonar tool. Code coverage for each module is one of the significant metric that needs to be maintained.

Transition Aspect-Case 3: The assessment results showed that the Transition aspect of Case 3 is at Level 1- Ad-Hoc. Among fifteen practices, five practices are rated as fully achieved, six practices are largely achieved and five is partially achieved. Transition aspect of project #3 is the one that needs most improvement among other aspects. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Transition Aspect-1st Level:

Code is under configuration control. There is no multiple branches in the system. All development is conducted over a single mainline. Check-in, check-out mechanisms are being used. Code is labeled and versioned with every change. Code cannot be committed without commenting. Change sets in SVN are linked to the issue items on Jira that means the reason to change code is clear.

Code sets are being integrated in the "mainline" environment after they are developed in developers' local computers. Developers are encouraged to commit at least once a day. However, this depends on the developer. New development type issues may take to integrate longer. At this level, our suggestion is to perform physical code configuration control to audit the changes in the code before the build.

Deployment of the solution: The code in mainline is compiled and sent to the testers two times a day. Code is automatically deployed to different environments with a single operation.

Testing the integrated solution: Test activities start after coding activities are finished. Testers start to write test cases after that time. Manual functional tests are run by testers. Manual start-up tests are run after deployment.

Teams are informed about the situation of transition activities with communication.

Supporting documentation are developed and published to the customers on website.

Transition Aspect-2nd Level:

Transition aspect activities are performed iteratively and incrementally. Product owner involves in acceptance when he needs. At the end of every iteration product owner are needed to be involved in acceptance activities (i.e. sprint review meetings). Test and development team work at different rooms that needs to be rethought.

There is a significant problem in balancing the flow of the work in transition activities. Testing process starts right after the coding activities are completed. However, testers are needed to be involved in the development process at earlier stages. (Parallel to coding activities). Writing automated tests is suggested to achieve the balance and improve code quality. Efficiency of testing activities are needed to be elaborated more.

Transition Aspect-3rd Level:

Code is not continuously integrated. Automated functional test suite does not exist. Transition activities are needed to be made visible to everyone. Tools may be utilized for this purpose. Automated integration tests are needed to be developed. Automated functional test suite are needed to be developed and improved continuously. Unit tests and automated test are needed to be run with every commit.

The status of the code base are needed to be tracked over tools. Tools are needed to be utilized for continuous integration and test automation.

In order to improve collaborative working, we suggest to gather test and development team members in the same office. Improvement actions are needed to be taken to improve transition activities. We suggest performance of retrospective studies regularly. Metrics are needed to be collected to observe the problematic areas of transition aspect.

Management Aspect-Case 3: The assessment results showed that the Management aspect of Case 3 is at Level 3- Effective. Among eighteen practices, thirteen practices are rated as fully achieved, five practices are largely achieved. Below we present our finding, the positive and negative evidences related to the practices and improvement suggestions.

Management Aspect-1st Level:

Project #3's vision was defined in strategy meetings, revised every year and known by every employee in the team. Strategic decisions about the products are stored in Jira. These records are also updated when the development related to these strategic decisions is completed for strategic decisions. Teams are formed based on the functionalities. People work dedicated to the project.

All internal stakeholders are aware of the agile values and principles. Agile trainings are performed to improve agile awareness among the team members. Offices are quite. Sport and music activities can be performed lunch time or after work.

Planning activities: The number of sprints and deployment dates are planned at the beginning of the year. Two weeks before the each sprint product development managers calculates the capacity of their teams and inform the product owner. Product owner prioritizes the backlog and decides which items to be included in the sprint based on the given capacity. Stand-up meetings are performed on weekly basis.

Teams perform story point estimation. Estimation is done at managerial level. Two development team leader and an analyst discuss about the size of the items. Test team leader does not involve these meetings. Progresses of the teams (velocity) are monitored daily through burn-down charts. Risk are evaluated as an internalized way. Preventive or corrective actions are taken. Risk monitoring approaches are needed to be evaluated. Risks may be tracked over Jira.

Management Aspect-2nd Level:

Management aspect activities are performed iteratively and incrementally. Teams effectively communicate in planning and estimation activities. Managerial tasks or documents do not prevent and cause lateness in development activities. Sprint planning activities are performed two weeks before the start of each sprint. Informal procedures are applied for the approval of management decisions.

Management Aspect 3rd Level:

Planning and monitoring are performed continuously. Size and effort are estimated. However, estimations are made managerial level. Developers' opinion are needed to be obtained for effort estimation. Test team leader may involve the estimation meetings of development teams to obtain idea about the items. Jira tool is utilized for the conduct of management aspect practices.

Management aspect's activities are followed through measures. Deviations between planned vs actual effort and size are two of them. Velocity of the team is monitored instantaneously and corrective actions are taken when a problem is observed.

4.2.2.4. 4th Case Study: Organization I, Project #4

Organization I is a solution provider for information and communication technologies in local and global market. It is located in Ankara, Turkey. Due to confidentiality issues, the name and the scope of the assessed project are kept private. We name the assessed project as "Project #4". Project #4 includes 4 software developers, 1 test engineer, 1 product manager and 3 part-time business analysts. There is no specific project manager in the project.

The project started in April 2014. Since the beginning of the project, 600KLOC has been developed. The product is being developed on PHP language. Iteration length varies between 15 days to 30 days. There is no consistency in iteration lengths. This is a

software product which internal dynamics had impact on the product idea and there is no external customer.

In the scope of this case study we assessed aspects of project #4 through interviews. Because of high confidentiality issues, we couldn't able to observe the direct evidences from the project. We performed the assessment in three-hour time with a product manager, a software team leader and a test engineer one by one. Total effort for appraisal is 6 hours.

The functional domain of the assessed project, is classified as the "Information System" based on CHAR group method [96].

Findings of Case Study 4: Figure 16 gives the colored schema of the assessment ratings to capture the detailed results at a glance. Each column refers to the practices of AgilityMod. The color codes and numbers in each cell refer to the achieved levels of these practices.

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	3	2	3	2	2	2	-	-	3	2	2	1	2	3	3	2	1	1
CONSTRUCTION	2	2	2	2	-	-	-	-	3	3	2	2	1	2	3	2	1	1
TRANSITION	3	3	3	2	2	2	-	-	3	3	1	3	1	3	3	2	1	1
MANAGEMENT	2	3	1	3	3	1	2	2	3	3	3	2	0	3	3	2	1	1

Figure 16 Rating of Each Practice for Case 4

Project team prefers to adapt agile practices into their processes to obtain rapid feedback for a novel project. Two members of the team had formerly worked on agile projects. Other members are new to the agile processes, values and principles. At the beginning they agree on the iteration length and start with some of the scrum practices. One of the major problems in project #4 is that a common perception among team members for agile principles and values is not established. The appraisal results showed that the best-applied aspect is Construction which is at the second agility level: Lean. Exploration and Transition aspects are at Ad-Hoc level. Finally, the Management aspect is at Not Implemented level. Figure 17 displays the achieved level of each aspect in the bar chart view.

Figure 17 Rating of Each Practice of Case 4

Below, we specify the assessment findings based on AgilityMod_v3.0 for each aspect and each agility level. The full assessment report is provided in [97]. .

Exploration Aspect-Case 4: The assessment results showed that the Exploration aspect of Case 4 is at Level 1- Ad-Hoc. Among sixteen practices, four practices are rated as fully achieved, nine practices are largely achieved and three is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Exploration Aspect - 1st Level:

Capturing customer and business needs: The project has emerged from the company's interior dynamics, and the software requirements are determined by the development team. Once the requirements are formed, they are defined as "epic" on the Jira tool.

In terms of elaboration, the business needs which are defined as "epic", are detailed in the "user story" level. Use case is also used occasionally in the project. However, we couldn't evaluate the adequacy of the defined units since we are not allowed to see original evidences. Therefore the compatibility of the requirements with INVEST (Independent, Value-Added, Small, Estimable, Testable) criteria should be evaluated. The non-functional system requirements are needed to be defined on the system as epics or user stories.

In case of incompatibilities in requirements, the team communications over Jira tool, although they are in the same room. Communication channels are used as required. Dependencies among requirement items are evaluated based on personal experiences. There is a need to specify functional dependencies among the requirement items.

Managing the requirements artifacts: Backlog is present on Jira. The specifications to be developed are classified in predefined phases. The value of the item plays an impact on the prioritization of the backlog items. Prioritization is done on the beginning of each sprint. Whenever a change on the existing requirements is defined, a new issue is created on Jira for major changes; the existing issue is updated for minor changes. The created issue is assigned to the "scrum master", and the scrum master directs the issue to the related developer. The scrum master is undertaken the role of the product owner. However, it is advised to avoid this role slippage in Scrum method. When a change on requirements is required on any stage of the process, we suggest a quick review of the change with whole team face to face rather than discussions conducted on the tool. This way, it would be much more efficient and effective than being informed from the system.

Team uses the KanBan board view in Jira. In addition to KanBan board, the backlog that is ordered by priorities are needed to be made visible.

Exploration Aspect – 2nd Level:

Exploration aspect activities are performed iteratively and incrementally. However, iterations are not periodic. Whole team members work in the same physical environment. Daily meetings are only performed among developers, business analysts should also participate in these meetings to improve collaboration and communication.

The requirements are prepared in the preceding sprints. Requirements are peer-reviewed and confirmed using Jira tool. However, it was mentioned that the requirements or change requests may wait long on Jira until being confirmed. It is possible that a change request may be left unnoticed. In order to speed up the process, effective communication channels need to be used.

Exploration Aspect – 3rd Level:

Incorporating engineering practices and tools into exploration aspect activities: Requirements are sometimes defined as user stories, and sometimes as use cases. Consistency needs to be maintained on the requirement forms. Jira tool is being used for tracking and managing the requirements.

Business analysts are responsible from requirements development and analysis activities. There is a need to improve the collaboration and shared responsibility with other team members for exploration aspect activities.

In case of a mistake made by a team member, a light penalty method is applied. Precautions are taken to prevent the occurrence of the mistake again, after resolving the issue. In agile methodologies, rather than penalizing the people, the root-cause analyzes should be done to investigate the cause of the mistake and precautions should be taken to improve the process accordingly. Conduct of regular meetings on process evaluation provide early detection of errors in the processes.

In organization I, there is no specific carrier plan for employees. Individuals learn and improve by their own efforts. An organizational training strategy needs to be developed. Lessons learned for exploration aspect need to be recorded. We suggest the establishment of a knowledge platform for such purposes. All team member especially new comers to the team need to be trained about agile principles and values and the way the things are done in project #4.

Exploration aspect needs improvement. It is suggested to perform retrospective meetings after every sprint to evaluate the positive and negative implementations. In order to specify the problematic areas well, we suggest the establishment of a measurement infrastructure.

Construction Aspect- Case 4: The assessment results showed that the Construction aspect of Case 4 is at Level 2- Lean. Among fourteen practices, two practices are rated as fully achieved, nine practices are largely achieved and three is partially achieved. Below we present our finding, the positive and negative evidences related to the practices and improvement suggestions.

Construction Aspect - 1st Level:

The work items are elaborated and detailed by business analysts before the development. Mock-ups may be developed. All elaboration activities are done on paper. As the details about the requirements are revealed, they should be updated on Jira in order to form a base for the test activities. Testers complain about the insufficiency of requirement descriptions.

While exploring the design in sprint planning meetings, dependencies among design elements are taken into consideration. Personal experience of developers play a significant role in this evaluation. Alternative designs (we mean conceptual and detailed design) needs to be evaluated in these meetings. Establishing a dependency structure to observe the design elements affecting each other would provide much value to processes. We suggest evaluation of the impact of new requirement items on the system architecture. In addition, it is better to be sure that the design satisfies both functional and non–functional requirements in advance by using approaches like prototyping.

Coding is done using PHP. The change-sets are attached to the user stories or tasks on Jira. The reasons for changing the code are known with this approach. Coding standards needs to be clearly defined and applied.

Developers ensure the correctness of the code through manual graphical user interface tests. Code review is done occasionally. Effectiveness of these manual tests should be evaluated.

Construction Aspect 2nd Level:

Construction aspect activities are performed iteratively and incrementally. Whole team works in the same room, which enables the establishment of an effective face-to-face communication.

In terms of balancing predictive and adaptive work, code is reviewed occasionally before functional tests. We suggest the review of the code consistently on a regular basis. Static code analysis approach may be preferred for dynamic control of the code standards.

Approval and decision-making mechanisms for design and coding activities are operated informally. An effective mechanism for continuous evaluation of the dependencies between design elements should be established.

Construction Aspect 3rd Level:

Unit tests should be coded in the scope of developer tests. Test driven development approach may be adopted to obtain fast feedback and establish a more reliable system. Code needs to be refactored regularly. For continuous refactoring, the system should be supported with unit tests. Code and design should be updated especially after hot fixes to prevent technical dept.

GIT tool is used for code development and version control. “Crucible” tool can be used for peer code review. “Gliffy” tool can be preferred for design development. UML approach may be preferred for the design. These tools can work compatible to Jira.

There is collaboration among team members, however, no shared responsibility since each employee has a specific role and responsibility.

Experiences on the design and coding should be transferred into organizational knowledge.

Transition Aspect-Case 4: The assessment results showed that the Transition aspect of Case 4 is at Level 1- Ad-Hoc. Among fifteen practices, seven practices are rated as fully achieved, four practices are largely achieved and four is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Transition Aspect-1st Level:

Integration, deployment and configuration control: Development and test activities are performed in the same branch. Code's configuration control is established. Check-in, check-out mechanisms are being used. DB scripts and versions of 3rd party libraries are known for each release. When a code is changed, tasks are linked with change-sets on Jira. Therefore, code is changed in a controlled manner. However, especially for the correction of emergent bugs, code (change-sets) may be attached with unrelated items and changed without a control. This type of change may cause the deployment of code parts without testing.

Code pieces that are developed in parallel are merged in the development environment. Code is deployed to the production environment automatically, which is an aspect of PHP language.

Testing the integrated solution: Test activities start after coding activities are finished. Testers start to write test cases after that time. In cases of incomplete or ambiguous requirements, the developers and analysts (which is present at the same room), clarifies the requirements with face-to-face communication. Manual functional tests are run by testers. Test activities are not planned at the beginning of the sprints. Bugs found during the tests are recorded to Jira.

After the elaboration of the requirements, issues, epics, user stories are needed to be updated on Jira. Test activities are needed to be planned.

Making the progress visible: The state of the developed attributes (epic or user story) can be followed on the dashboard present on the Jira. However, the state of the integration activities, and the status of the mainline needs to be visible to all team members.

Transition Aspect-2nd Level:

Transition aspect activities are performed iteratively and incrementally.

Test preparation activities starts parallel to the coding activities. The changes on the requirements made after the establishment of the test cases lead to idle test cases, or repetition of the test effort. Requirements need to be stable within a sprint. Test cases should be updated after requirement changes. An infrastructure should be established where test cases can be updated easily on the system.

It is possible to eliminate the repetition of the manual tests by coding the unit test at the point where the bug is found. With this approach, not only the test coverage can be increased, but manual test can be avoided as well. (This is the leanest approach)

Transition Aspect-3rd Level:

Continuous integration approach needs to be used since code development and testing activities are performed in the same environment. Continuous integration (keeping the code in a working state, compilation of the whole application with every commit and running all the unit and automated tests at the background) should be performed after the system is supported with automatic tests and unit tests.

Tools are widely used for the conduct of the transition aspect activities.

Newcomers should be given process and field trainings. Experience on the integration and test areas needs to be transferred into an organizational knowledge center. Metrics on test, integration and deployment activities needs to be tracked. Deviations in these processes should be recorded and assessed.

Management Aspect-Case 4: The appraisal results showed that the Management aspect is at Level 0- Not implemented. Among eighteen practices, eight practices are rated as fully achieved and five practices are largely achieved, four practices are partially achieved, one practice is not achieved. Below we present the positive evidences obtained from the evaluation of this aspect and improvement suggestions for the practices.

Management Aspect-1st Level:

At the beginning of the project a feasibility study was performed and product vision was determined. Project vision should be updated regularly to in the light of technological improvements, etc. Documentation to be developed in the scope of the project should be determined.

An agile awareness is not established among the stakeholders. Agile perception is based on past experiences of the team members in agile software development. All stakeholders need to be informed about agile values and principles and a common view should be established.

Physical work space is compatible with the suggestions of the agile approaches. "Quiet working rooms" can be built in which team members can work when they need.

Daily activities are coordinated in the scope of "daily scrum meeting". These meetings are held only by software development team. There is a need of all team members join daily standup meetings.

Size and effort should be estimated for work products. An estimation approach should be determined. AE: story point estimation, cosmic measurement… If these metrics are defined on Jira, it would be easier to get the metrics (i.e velocity of the team)

Project progress is just monitored on a daily basis over the completed tasks. As the number of completed tasks is not a sufficient parameter, it would be better to monitor the velocity of the team, the size of the completed and remaining tasks. That would make the work visible.

Management Aspect-2nd Level:

Estimation, planning and monitoring activities are performed iteratively and incrementally.

Decision making should be done at the last stage as much as possible to wait until the variables are clear

Process assessment meetings needs to be made regularly, and the non-value added practices should be removed from the processes.

Management Aspect 3rd Level:

Agile approaches needs to be utilized for work effort and size estimation.

Jira tool is used for project management. Activities are followed on Kanban application of Jira.

Experience on the projects/product management should be transferred to an organizational knowledge center. A process measurement infrastructure should be established.

In the project the risk assessment activities are done for the hardware and software tools to be used. However, risk assessment is not performed on a regular basis.

4.2.2.5. 5th Case Study: Organization C, Project #5

Company 'C' is working on internet security domain. They develop products with the purpose of securing information on internet, securing websites and e-commerce applications and personal computers. 'C' is an international company, doing business over 100 countries, with its 25 million end users, and over 7000 business partners.

Product #5 is a digital advertisement sharing platform. It is in use and new versions of the product are being deployed continuously. The purpose of the project is to ensure the security of the advertisements and deliver harmless and focused advertisements to end users. The project includes 22 employees. There are 3 different development teams. On top of every team, there is a program manager, a scrum master for each of the teams, 4 testers, 13 developers and 1 architect. Apart from these members, there are product managers, which can be thought as product owners, living in USA and Turkey. Theta is built upon a legacy code. Java, PhP and Phyton languages are being used for different modules of the product. The project includes big data analyzes performed with the tools Cassandra and Hadoop. Scrum is being used for project management activities. Theta is built iteratively. Each iteration takes 3 weeks.

In the scope of this case study we assessed aspects of project Theta through interviews and direct observation of the evidences. The assessment was performed in three-hour time with the configuration manager and the quality assurance manager who is also scrum master. Interviews took 6 person-hours in total.

The functional domain of the assessed project, is classified as the "Complex Data-Driven Control System" based on CHAR group method [96]. The functional domain has been specified by program manager of the project #5.

Findings of Case Study 5: Figure 18 gives the colored schema of the assessment ratings to capture the detailed results at a glance. Each column refers to the practices of AgilityMod. Colors and numbers in each cell refer to the achieved levels of these practices.

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	3	3	3	3	3	3	-	-	3	3	3	3	3	3	3	3	3	3
CONSTRUCTION	3	3	3	3	-	-	-	-	3	3	3	2	2	3	3	3	2	3
TRANSITION	3	3	3	3	3	3	-	-	3	3	2	3	1	3	3	3	2	2
MANAGEMENT	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Figure 18 Rating of Each Practice of Case 5

Project #5 has been the one of the most successful projects based on the assessment results among six case studies. All of the practices of exploration and management aspects are rated as fully achieved. Construction aspect is at Level 3 by having only 3 largely achieved practices. The weakest aspect of project #5 is "Transition" aspect which is at level 2: Lean. The major improvement areas of this project are achieving continuous integration, continuous delivery and increasing unit test coverage and automated test ratio, refactoring continuously and managing technical dept better.

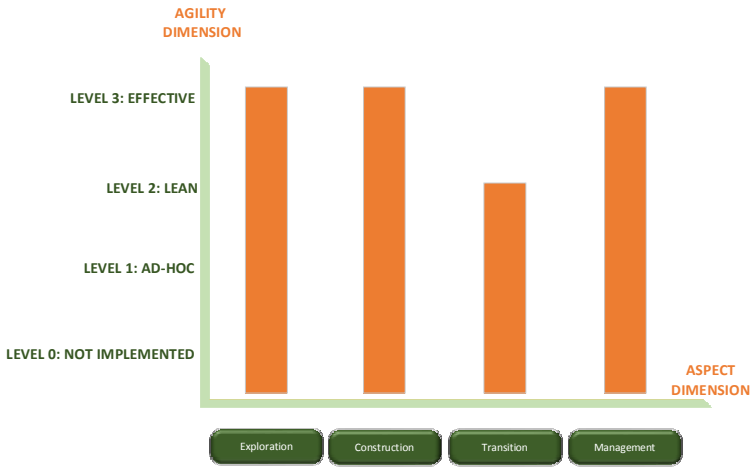

Figure 19 Achieved Agility Levels of Aspects for Case 5

We specify the assessment findings below for each aspect and each agility level. The full assessment report is provided in [97]. .

Exploration Aspect-Case 5: The appraisal results showed that the Exploration aspect is at Level 3- Effective. All of the sixteen practices are rated as fully achieved. Below we present the positive evidences related to the aspect practices and agility practices. The descriptions given below can be considered as good examples for an Effective level Exploration aspect.

Business needs are obtained from the product manager in USA over skype meetings. Product manager and product owners meet regularly. After the meetings performed over skype, product manager defines the business needs as "epics" in Atlassian's Jira tool.

Epics are elaborated in the scope of grooming meetings by teams in Turkey. Elaborated epics are recorded as "user stories" in Jira tool by keeping the relation and dependencies between epics and user stories. User stories are written in the form of a "real" user story. In addition, each user story includes the acceptance criteria in itself. Non-functional requirements are also transformed into user stories and test scenarios.

In addition, sprint planning meetings are performed. In sprint planning 1, "what analysis" is performed. Product owner and program manager involve in these meetings. In sprint planning 2: user stories are detailed and estimated, acceptance criteria are written. Estimation approach has an impact on the elaboration of the requirements. Until small numbered cards are opened, elaboration of the items continue.

Conflicts on requirements are resolved in sprint planning and grooming meetings where all team members are involved. Product owner from USA also attends these meetings over skype.

Dependencies among epics and user stories are established in Jira tool. Backlog is managed through grooming meetings, performed every week (apart from sprint planning meetings). Changes are discussed in a change control board. Every team member is notified about the change. Risk and impact analyzes are performed.

The approach of the team is not accepting a change in the course of a sprint. However, urgent or important change requests (i.e. coming from CEO) are handled. This is also an issue which is assessed in quality audits. If a new item is added to the sprint, a penalty point is given to the team. Backlog is prioritized based on the business value of the item and technical complexity. High valued items are developed in the first order.

Jira's dashboard view is being used to capture the situation of the ongoing tasks. The dashboard includes "backlog, new, in progress, done and verified" columns filled with issues in prioritized order. The product backlog and sprint backlog are also visible to everyone.

Exploration aspect activities are performed in an iterative and incremental way. Each

Communication interfaces are established between internal, external stakeholders. All team members come together and communicate on daily activities through daily stand-up meetings.

iteration is 3 month-long.

Frequent feedback is obtained from the customer. Whenever a major item is completed, a demo is displayed to the product manager. Team does not wait even 3 weeks to obtain feedback.

Communication matrix is kept to show the frequency of the communication between program managers in Turkey and product manager in USA. It is updated weekly. This approach aims to specify the relation between the requirement specification problems (decrease on specification of the items on the backlog is an indicator) and communication frequency.

Product manager involves sprint planning meetings. In addition, product backlog is groomed every week being independent from the upcoming sprint. This approach enables the achievement of the balance and flow of the requirements to the sprints without any interruption. Team spent two hours a week to elaboration activities. Since it is regular, the impact of this little time is significant.

Product features are reviewed by program managers and CEO over Confluence tool.

Quality manager performs process audits based on the metrics obtained from the tools (over an item list) and discuss the results with program managers weekly. Improvement actions are taken based on the feedback.

Apart from these, whole team come together at the end of every sprint in retrospective meetings. Feedback is obtained from the team. What is best about this process, action items are opened when a new issue is emerged and assigned to a person over the Jira tool.

All team members are responsible from exploration activities. There is collaboration and shared responsibility in the team. Sophisticated people are selected to the team. Issues are submitted to "issue pool", no assignment made except for the issues which require specialty. There is no specific project manager role in the team and no hierarchical structure. Teams are self-organized.

Continuous improvement is achieved in every cycle not only in project #5, but throughout the organization. All team members are trained when they come to the team for the first time about processes and the tools to be used (orientation training).

Exploration aspect activities are continuously monitored. Derived measures are calculated and displayed in the form of charts. All the measures are available through tools.

For example the increase rate of the items on the backlog is monitored through cumulative story diagrams.

Construction Aspect-Case 5: The appraisal results showed that the Construction aspect is at Level 3- Effective. Among fourteen practices, eleven practices are rated as fully achieved and three practices are largely achieved. Below we present the positive evidences obtained from the case study and improvement suggestions for the practices that are largely achieved.

Construction Aspect - 1st Level:

For *the elaboration of work items*, team had meetings with product manager or program manager. Workflow diagrams are developed. UML diagram types are utilized. Team ensures that the design of the product meets the non-functional requirements through the proof of concepts. Alternative design solutions are evaluated before the development.

PHP, Phyton and Java programming languages are used for coding. Cassandra and Hadoop applications are used for business intelligence studies. Coding standards are

applied to improve the code quality. The system does not allow check-out of the code without writing a comment. The efficiency of the code comments are needed to be evaluated. This will increase the clarity of the code especially at the maintenance phase.

Developers tests the correctness of the code through unit tests and manual console tests.

Construction Aspect 2nd Level:

Coding activities are performed iteratively and incrementally. All members within a team come together and communicate on daily activities through daily stand-up meetings. Design decisions and code parts can be discussed and reviewed over Crucible tool. Everyone can comment on the code part and all comments are seen by other reviewers. After the review, final remarks can be done with a meeting.

Team balances the adaptive work and predictive work for design and development activities. Risk evaluation is performed in a formal way. Prototyping and proof of concept approaches are performed to quickly observe if the suggested solution works or not. Static code analysis to check the conformance to coding standards are performed regularly.

Dependencies between design elements, needs to be specified in a higher abstraction level. Proper abstraction level for dependency analysis are needed to be specified. A matrix structure are needed to be developed to view the impact of requirements on design elements and the relations among design elements themselves. An efficient approach to evaluate dependencies among design elements should be defined and decided approach are needed to be applied the whole system. (such as Dependency Structure Matrixes)

Construction Aspect - 3rd Level:

Team applies the rules of coding standards. Code is reviewed by peers. Test cases are also reviewed by peers. In addition to code reviews, design is reviewed.

The coverage ratio of the unit tests needs to be increased.

There is ability to manage technical dept to some level. Static code analysis and code reviews may be utilized to identify technical depts. Technical search activities are performed at a high component level. The impact of new arriving requirements on modules and a lower level module components are evaluated based on personal experiences. In some cases, team needs to develop quick solutions or hot fixes which may result in technical dept. Recovering from the technical dept is the responsibility of the developer who develops it. There is no specific mechanism to control it.

Code is refactored when needed. The next step should be continuous refactoring with the support of unit tests and automated integration tests. We suggest review of risky design parts. Refactoring issues might be opened after the reviews.

GIT tool and check-in, check-out mechanisms are being used for version control. “Sonar” tool is being used for static code analysis. “Gliffy” tool is preferred for design development.

Code quality metrics are kept and evaluated through Sonar tool. Code coverage for each module is one of the monitored metrics.

Transition Aspect-Case 5: The appraisal results showed that the Transition aspect is at Level 2- Lean. Among sixteen practices, twelve practices are rated as fully achieved and three practices are largely achieved, one practice is partially achieved. Below we present the positive evidences obtained from the evaluation of this aspect and improvement suggestions for the practices.

Transition Aspect-1st Level

Code is under configuration control. There are multiple branches: "development", "release branches" and "master branch". Check-in, check-out mechanisms are being used. Code is labeled and versioned with every change. The relation between the change-sets and the Jira items are established through commenting. Jira item number is added to the comment while check-out the code. Code cannot be committed without commenting.

Code sets are being integrated in the "development" environment after they are developed in developers' local computers. Developers are encouraged to deliver the code to the "development" branch at least once a day. The works that exceeds two days delivery are labelled and the reasons are investigated. Physical code configuration control may be performed to audit the changes in the code before the build.

Build is taken automatically. It may be manual for necessary conditions. Deployment is performed automatically over scripts. Build errors are checked before deploying the solution (rebuild tests). Posts tests are run after the deployment to detect the errors.

Test cases are written based on acceptance criteria attached each of the user story. Performance and usability tests are performed in terms of the test of the non-functional requirements. Tests are run in two phases: informal and formal. In informal phase, that is the state in which all the tasks related to a user story is open, no bug is recorded during the tests, instead, the found bugs are added as comments into the user stories. In formal phase (2 days left to the deployment) user acceptance tests and regression tests are run and bugs are recorded.

Transition activities are made visible over Jenkins tool. The status of the code base can be tracked. Documentation decisions are taken at the beginning of the project and supporting documentation are developed using confluence tool.

Transition Aspect-2nd Level:

Transition aspect activities are performed iteratively and incrementally. Testers are involved in the process at the planning phases. While developers writing codes, testers write test cases. Test cases are also reviewed. Studies related to establishing a continuous integration and continuous delivery have been started. The balance will be more achieved with continuous testing process.

Informal procedures are applied for the approval of transition decisions. Non-value added activities are eliminated from the processes through retrospective meetings.

Transition Aspect - 3rd Level:

Team members integrate at the least once a day to the development branch. However, since the infrastructure is not complete, the benefit is not fully achieved yet. We suggest improving the automated test suites and running all of them every night. At the best scenario, we suggest running no manual tests except for the exploratory tests.

Tools are effectively utilized for Transition aspect activities: Jenkins tool is used for continuous integration and batch processing. GIT tool is used for version control. Selenium tool is utilized for development of automated tests. Confluence tool for information sharing.

Transition aspect's metrics can be collected and monitored over Jenkins tool. For example, the number of the successful and failed builds.

Management Aspect-Case 5: The appraisal results showed that the Management aspect is at Level 3- Effective. All of the eighteen practices are rated as largely achieved. Below we present the positive evidences obtained from the evaluation of this aspect. The descriptions given below can be considered as good examples for an Effective level Management aspect.

At the beginning of the project a feasibility study was performed. Product vision, roadmap and technical challenges are specified. For the documentation of vision and scope one page project data sheets can be preferred.

While forming the team, technical knowledge and experiences of people are taken into account. Not all the team members have same level of experience. All team has a technical background and have capability of involving coding activities.

Internal and external stakeholders are aware of the agile values and principles. High-level managers are one of the major drivers of agile adoption. They involve in the processes. Agile awareness of each team member is assessed periodically. Training needs are identified from the results of these assessments.

Planning is based on team velocity and story points and done in sprint planning 1 and 2 meetings. Team does the planning itself. Since the backlog is estimated based on story points and the velocity of the team is known, team knows how many sprints are needed to complete backlog items. Daily activities are coordinated through daily stand-up meetings within the teams. Scrum of scrum meetings are conducted among the teams in which the scrum masters and product managers involved.

Teams perform story point estimation and applies poker planning approach to estimate the job to be done within a sprint. Story points (completed vs remaining) are updated regularly through daily standup meetings.

A well-structured monitoring system has been established. The purpose of the quality management team is to establish a platform that is measurable, repeatable, and

analyzable. Various charts are utilized for monitoring. The whole team is informed about the progress and the results.

Project risks are tracked regularly. Regular meetings are performed for risk management. Program managers enters and monitors the risks. Change on the risks of the project are reported at the end of every iteration. Risk parameters are also maintained.

Management aspect activities are performed iteratively and incrementally. Internal communication frequencies are monitored through communication matrixes. Since communication is a key issue in the project, whenever a problem occurs, first the communication quality is checked. Planning and estimation activities are performed continuously. Regularly retrospective studies are performed and corrective and improving actions are being taken. Non-value added activities are removed from the processes. Management aspect activities are being improved continuously. For example team decides 3 weeks iteration length after trials of two and four weeks.

A whole team approach is applied in estimation.

People share their knowledge and learn from each other. In addition to the retrospective meetings and actions are conducted in the organization, lessons learned are needed to be kept and updated regularly at the end of the projects.

4.2.2.6. 6th Case Study: Organization NT, Project #6

Organization NT develops products in the field of information and communications technologies. It is a research and development company. It was ranked 2nd in telecom sector with its 34 patent applications in 2013 in Turkey. NT trades on Borsa Istanbul (BIST).

Project #6 is a WebRTC (web real time communication) gateway project. It enables voice and visual communication between via web browsers. Project team includes 45 people divided into 6 scrum teams. The product owner and customer are in USA. Product owner involves in meetings via teleconferencing. Team include developers, testers, scrum masters and architects. The architects also work as business analysts. In USA, there are 2 solution architects who are in communication with the product owner. On top the all scrum teams, there is a technical project manager.

Since the beginning of the project, 2012, 500KLOC has been developed. Each iteration is 3-week length. There is signed contract between the organization and the customer specifying the dates and budget.

In the scope of this case study we assessed aspects of project #6 through interviews and direct observation of the evidences. The assessment was performed in four-hour time with the quality manager, test manager, solution architect and technical project manager. Interviews took 12 person-hours in total.

The functional domain of the assessed project, is classified as the “Complex Control System” based on CHAR group method [96].

Findings of Case Study 6: Figure 20 gives the colored schema of the assessment ratings to capture the detailed results at a glance. Each column refers to the practices of AgilityMod. Colors and numbers in each cell refer to the achieved levels of these practices.

	1. AD-HOC								2. LEAN				3. EFFECTIVE					
									Iterative		Simple		Technically Excellent		Learning			
Aspects/Practices	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	GP 2.1.1	GP 2.1.2	GP 2.2.1	GP 2.2.2	GP 3.1.1	GP 3.1.2	GP 3.2.1	GP 3.2.2	GP 3.2.3	GP 3.2.4
EXPLORATION	3	3	3	3	3	2	-	-	2	2	2	2	2	2	2	3	1	2
CONSTRUCTION	3	3	3	3	-	-	-	-	3	3	2	2	1	3	3	3	2	2
TRANSITION	3	3	3	3	2	3	-	-	2	2	3	2	1	3	2	3	1	1
MANAGEMENT	2	3	2	3	2	3	3	3	3	2	3	2	1	2	3	3	2	1

Figure 20 Rating of Each Practice of Case 6

The assessment provides very promising results for Project #6. All of the aspects meets the requirements of Level-2: Lean. Project #6 has been adopting agile processes for 2,5 years. One of the major challenges for them is the change of product owner approximately with every 6 months. They are working with the 4th product owner nowadays. Adaptation problems come with the change, since the agile knowledge of product owners vary. The team is good at applying scrum procedures and requirements elicitation activities. The major improvement areas are as follows: performing regular assessment activities to improve the aspects and establishing a comprehensive and observable metric structure, adapting engineering practices such as continuous integration and unit testing, establishing agile culture among team members and support face to face communication. Figure 21 displays the achieved level of each aspect in the bar chart view.

Figure 21 Achieved Agility Levels of Aspects for Case 6

We specify the assessment findings below for each aspect and each agility level. The full assessment report is provided in [97]. .

Exploration Aspect-Case 3: The assessment results showed that the Exploration aspect of Case 6 is at Level 2- Lean. Among sixteen practices, six practices are rated as fully achieved, nine practices are largely achieved and one is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Exploration Aspect - 1st Level:

Capturing customer and business needs, elaborating requirements: Business requirements are obtained from product line manager (product owner) who lives in USA via teleconferencing. Product owner specifies the needs in VersionOne tool as "epics". Architects in Turkey and the solution architects in USA discuss the epics and transform the epics into user stories. User stories are still at a high level at this phase. The nature of the software product #6, requires the identification of the message flows between different server and clients running on different protocols. Architects in Turkey also specify these call flows. Call flows are also essential for testers to write test cases. However, in some cases, specification of the call flows may be finished at the end of the second week of a sprint that is quite late for the start of test activities.

Resolving conflicts: Conflicts on requirements are resolved through communication. Solution architects in USA are asked to clarify the requirements. Team members (developers and testers) prefer to communicate face-to-face and email based communication. Face to face communication change needs to be used mostly rather than other approaches.

The relations between epics and user stories are defined in VersionOne tool.

Managing the requirements artifacts: Three backlogs are maintained in the project: the release backlog, the product backlog and the sprint backlog. Release backlog includes the items to be developed within the next 6 months. The sprint backlog includes the items to be developed within the sprint. Prioritization of the backlog items are done by product line manager. Small sized change requests are accepted during the sprint. The large ones are mostly rejected or replaced with the low priority items in the backlog. Scrum masters are involved in the meetings where changes are discussed, then scrum masters transmit the information to scrum teams.

Making the artifacts visible: Backlogs are maintained in VersionOne tool and visible to everyone in the team. We suggest using dashboards to capture the ongoing situation of the tasks at a glance.

Exploration Aspect-2nd Level:

Exploration aspect activities are performed in an iterative and incremental way. The items for the upcoming releases are specified before the start of the release (a six-month work) while other development activities are conduct for the current release. The iteration length needs to be consistent.

Product owner or solution architects on behalf of product owner involve in sprint planning meetings to specify the epics and the alternative solutions. User stories are elaborated in sprint planning meetings where team members are involved. After the release backlog items are specified in the previous months, user stories are detailed within each sprint in sprint planning meetings. This may prevent moving to the coding phase as quickly as possible. We suggest elaborating epics and user stories in the previous months. Functional design documents needs to be developed regularly and earlier in a sprint.

In terms of specifying the non-value added activities in aspects, we expect teams perform regular retrospective studies and eliminate non value added activities. In the project,

retrospective studies are performed in long periods. We suggest evaluating the way of doing works at the end of each sprint.

Exploration Aspect-3rd Level:

It needs to be ensured that user stories are in sufficient level of detail for developers and testers to conduct their work. Properness of the user stories to INVEST criteria (Independent, Value-Added, Small, Estimable, Testable) needs to be evaluated. Functional dependencies among epics needs to be specified.

There are issues related to collaboration and shared responsibility in the team. Teams are semi-self-organized. There is a technical project manager role in the project. Scrum masters play significant role in the conduct of activities. Pooling approach may be enhanced. Approaches of scrum masters differentiate from team to team which cause motivation or demotivation of team members. When a problem or error occurred, team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error. No-one is blamed.

Project team members take trainings on general agile concepts. Specific trainings would increase the adoption of agile culture better. Lessons learned are needed to be updated regularly at the end of each release.

We suggest conduct of quality audits to encourage people for improvement and observe problematic areas for exploration aspect.

Story points related to each user story are kept and monitored. Metrics needs to be calculated and monitored at the end of each sprint. Metrics needs to be shared with whole team members to involve them to the processes and discuss the improvement suggestions.

Construction Aspect- Case 6: The assessment results showed that the Construction aspect of Case 6 is at Level 2- Lean. Among fourteen practices, nine practices are rated as fully achieved, four practices are largely achieved and one is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Construction Aspect - 1st Level:

Requirements that need further elaboration are detailed in design spikes. In addition, user stories are discussed in sprint planning meetings. Developers may ask for clarification of the user stories in these meetings. Call flows are specified by software architects. Alternative solutions are evaluated with solution architects. Sequence diagrams are developed. It was mentioned that issues related to the development of the sequence diagrams on time even if high level design tasks for the development of the call flows are created in relation to epic or user stories.

Java, C++ and C is utilized for coding server side and Java script, CSS, J query, and HTML5 is utilized for client side. Coding standards are partially applied. The system does not allow check-out of the code without writing a comment. However, comments are not

reviewed. We suggest evaluation of the code comments to increase the clarity of the code especially at the maintenance phase.

Developers tests the correctness of the code through manual tests on their own branches (labs). Code is reviewed by peers or architects or scrum masters. We suggest creating tasks for code review activities.

Construction Aspect - 2nd Level:

Coding activities are performed iteratively and incrementally. Team members come together and communicate on daily activities through daily stand-up meetings. However, consistency needs to be established in the conduct of these meetings. Software architects work closely with developers.

Proof of concepts are used to be in the safe side for design decisions and to quickly observe if the suggested solution works or not. The flow to design documents to test teams needs to be improved. Static code analysis to check the conformance to coding standards needs to be performed regularly.

Construction Aspect - 3rd Level:

Coding standards needs to be specified and reviews need to be performed to check the conformance to coding standards. Unit test needs to be developed. Code needs to be refactored continuously. Ability to manage technical dept needs to be improved.

SVN and Clear Case is used for version control. Issues are tracked on Jira. A tool is utilized for design review and code review activities. Tool support might be used for static code analysis.

Collaboration among team members are seen in development activities. Developers mostly select their own tasks to work except for the ones require specialties.

Construction metrics needs to be evaluated at the end of each sprint. Sonar tool would be helpful. Code complexity, code coverage might be monitored.

Transition Aspect-Case 6: The assessment results showed that the Transition aspect of Case 6 is at Level 2- Lean. Among fifteen practices, eight practices are rated as fully achieved, five practices are largely achieved and three is partially achieved. Below we present our findings, the positive and negative evidences related to the practices and improvement suggestions.

Transition Aspect-1st Level:

Code is under configuration control. Core side of the code is developed on Clearcase and Client is on SVN. Check-in, check-out mechanisms are being used. Code is labeled and versioned with every change. Code cannot be committed without commenting. Changesets in SVN and Clearcase are linked to the issue items on Jira that means the

reason to change code is clear. Physical code configuration control may be performed to audit the changes in the code before the build.

Code is either deployed automatically or manually based on the condition of the environment to be deployed. Internal deployments are performed automatically. Quick tests are run after deployment.

Functionality, sanity, integration, migration, regression and robustness tests are performed. They are mainly white-box type tests. Whenever a bug is found, an issue is opened in Jira and assigned to the developer.

Transition activities needs to be made visible.

Transition Aspect-2nd Level:

Transition aspect activities are performed iteratively and incrementally. However, the iteration length is not consistent. It changes between 3 weeks to 5 weeks. Consistency in the length of the sprint needs to be established. When the iteration length are increased and tests are left to later phases, the process turns to mini-waterfalls.

Although the team members share the same office, face to face communication is not used effectively between test team and developers.

Tests manager involves in the development process at the release planning phase. The timeline is planned along with the technical project manager and product owner. However, testers are involved at later phases, at the end of the 2nd week of a 4 week sprint. In the 1st week testers are involved in daily scrum meetings. Testers may wait functional design documents covering the call flows that are inputs for the test cases. In order to resolve this problem, functional design documents needs to be delivered earlier days of a sprint.

Test cases are reviewed and approved by software architects and product owner. Affected test cases from changes are updated and approval procedure on HP Quality Center is repeated. Informal procedures are applied for the approval of transition decisions.

Transition Aspect-3rd Level:

Code is integrated in the mainstream at the end of the sprints that means a significant workload because of merging. Code needs to be integrated continuously.

The ratio of automated tests are increased with every sprint. Currently %35-%40 of all tests are automated. The retest cost of a bug might be eliminated by writing the unit tests and the automated test where the bug is found. By this way the coverage ratio of the software is increased at the same time. Tests needs to be run continuously. Currently testing starts after all development is finished. At the best case we suggest conduct of no manual tests except for the exploratory tests.

SVN, ClearCase, CruiseControl are utilized for the conduct of transition aspect practices. Tools are integrated into testing activities. Rannorex tool is utilized for development of automated tests.

In terms of establishing a learning structure throughout the project the collaboration between test team members and developers needs to be improved. Everyone needs to work towards a common goal. Everyone needs to feel his/her value to the shared goal.

Root cause analysis for the found defects at different phases of the life cycle needs to be performed and feedback needs to be given to developers to prevent future errors. This analysis needs to be performed at the end of each sprint. The reasons of external and internal defects needs to be analyzed. Metrics needs to be shared with whole team members to involve them to the processes and discuss the improvement suggestions.

Management Aspect-Case 6: The assessment results showed that the Management aspect of Case 6 is at Level 2- Lean. Among eighteen practices, nine practices are rated as fully achieved, seven practices are largely achieved and 2 practices are largely achieved. Below we present our finding, the positive and negative evidences related to the practices and improvement suggestions.

Management Aspect-1st Level:

For each release, release scope is specified at a high level. At the beginning of the project we suggest identification of project vision, establishment of a shared understanding for vision.

While forming the team, it was one of the critical factors that everyone can involve coding activities.

Internal stakeholders are aware of the agile values and principles through internal and external agile trainings. Although the product owner is aware of how the project is conducted based on agile rules, he may sometimes ask for new requests that conflicts with agile principles. We suggest the evaluation of the effectiveness of the agile trainings taken. Based on the results improvement action may be taken and agile awareness are established in a better way.

Offices are open and sometimes, the environment can be very loudly. However, there are small offices for two or more people work in quietness. There is enough air and light in the environment.

In terms of planning release plans and sprint plans are prepared. Both release backlog and sprint backlog is maintained. Since the backlog is estimated based on story points and the velocity of the team is known, team knows how many sprints are needed to complete backlog items. Daily activities are coordinated through daily stand-up meetings within the teams. However daily standup meetings are not performed regularly.

Scrum of scrum meetings are performed to ensure the communication among development teams. However, product owner might give promises to customer to develop specific features on a specific date without asking the project team in Turkey and without performing any estimation. This situation is creating significant burden

development teams and may cause overtime work. The impact of the product owner's promises to the morale of development teams and the timeline needs to be discussed with product owner.

Teams perform story point estimation. However, the estimation is performed by product owners and architects. For the user stories that is bigger than 40 story points, design spikes are performed.

Progress of the teams are monitored through daily scrum meetings and scrum of scrum meetings conducted twice a week. Sprint and release burn down charts are discussed in these meetings. Project technical manager also maintains the resource allocation.

Project risks are tracked regularly. When a risk realized, a person is assigned to it, mitigation plans are defined and reported to upper-level management in monitoring meetings. The risks are maintained in excel sheet or power point presentations.

Management Aspect-2nd Level:

Management aspect activities are performed iteratively and incrementally. Communication interfaces between scrum masters and team, among scrum masters; among scrum masters and high level managers are established. This thinks a staged hierarchical communication interface. Other team members need to be involved in planning and estimation activities.

Managerial tasks or documents do not prevent and cause lateness in development activities. Approval of the managerial documents do not cause lateness in development activities. The improvement items specified in retrospective meetings, needs to be turned action items and be assigned and tracked regularly.

Management Aspect 3rd Level:

A whole team approach needs to be applied for estimation to improve the collaboration and increase the accuracy of the results. The size of the test items needs to be specified and impact analysis needs to be performed for change requests.

VersionOne and MS Excel tools are utilized for the conduct of management aspect practices. A more effective tool might be selected for the management of aspect activities.

Leaders are also part of the activities in project. People share their knowledge and learn from each other. New comers to teams take a 4 week training on technical topics.

Collected metrics are shared and discussed at managerial level, whole team members needs to be involved in the discussions.

4.2.3. Assessment Validation

We aimed to achieve the validity of the case study results by discussing our findings and observations with interviewees and managers in the organizations. Since there is no

commonly accepted or well-structured another Software Agility Assessment Reference Model in the literature, we couldn't compare our results to the others as a way of ensuring validity of the model. Instead, we prepared the agility assessment reports provided in Appendix B for Case 1 and in technical report for Cases 2-3-4-5 and 6, after each case study was conducted. We shared these reports with case organizations. In addition, we presented the results to the assessment teams, in some cases to managers and CEOs of the organizations. Presentations covered the assessment findings, levels of aspects and improvement suggestions. After or during the each presentation we discussed the results with attendees. At the end of each presentation we asked them to fill the questionnaire given below. All the questions except for the last one in the questionnaire are designed as open-ended.

The purpose of the questionnaire is to obtain the opinions of the people who attended the assessment process if we could capture the problems in the aspects, bring new improvement opportunities in terms of agility. By this way we aimed to validate both the assessment findings in each organization and AgilityMod.

Table 10 Questions in Validation Questionnaire

ID	Question
1)	What is your role in the organization? Could you please describe your background and experiences on agile software development?
2)	Does the report/presentation cover all the improvement areas that you notice about the organization's agile processes? If not, what are the missing ones?
3)	Which of the findings and improvement suggestions presented in the report/presentation have you noticed before? Which of them were new to you?
4)	Does the agility improvement path that is presented to you in the assessment report sound reasonable? Would you prefer the same improvement path? What would be your priorities?
5)	To what extent the presented findings and improvement opportunities in your projects overlap? Please select the scale that applies.

For the 5th question, we asked attendees to write the most applicable scale for each of the aspects. We used a four level scale to express opinions of representatives' "Not Achieved"- "Partially Achieved"- "Largely Achieved"- "Fully Achieved".

Below, the opinions of the interviewees are presented for each case. Each comment should be evaluated within its specific context.

Case 1: Organization NM, Project #1: To remind, project #1's aspects agility level are Level 1: Ad-Hoc, Level 1: Ad-Hoc, Level 0: Not Implemented, Level 0: Not Implemented for Exploration, Construction, Transition and Management aspects respectively.

It was specified that presentation both covers previously discovered and undiscovered improvement items from an agile perspective. They mentioned that the results are very beneficial for them to discover their potential for improvement. The suggestions that were given in term of construction and transition aspects made the most influence for them. As an answer of our 4th question they specified that even though the suggestions are meaningful and rationale to apply, they require significant changes in infrastructure and time and budget. Therefore they mentioned the need for an additional consultancy and a detailed improvement plan.

Three people answered the questionnaire and following pattern is realized for the 5th question. All of the attendees selected the highest scale.

Table 11 Ratings of the Findings-Case 1

Aspects	Project Manager 1	Project Manager 2	Software Team Leader
Exploration	Fully Achieved	Fully Achieved	not rated by him
Construction	Fully Achieved	Fully Achieved	Fully Achieved
Transition	Fully Achieved	Fully Achieved	Fully Achieved
Management	Fully Achieved	Fully Achieved	not rated by him

Project managers mentioned that the findings and improvement suggestions presented in the report are beyond their expectations and provided new viewpoints to them. The software team leader did not involve on the discussions of exploration and management aspects and did not give ratings for them. He mentioned that we called attention to major problems in their processes.

Case 2: Organization G, Project #2: Project's aspects agility level are specified as "Level 3: Effective" for all the four aspects. Although all the aspects are labeled as Level 3, some of the aspect and agility practices are rated as largely achieved.

We presented the assessment findings to one of the software team leaders in Project #2 who is also responsible for configuration management and system infrastructure activities. His ratings for the aspect based findings are as follows:

Table 12 Ratings of the Findings-Case 2

Aspects	Software Team Leader
Exploration	Fully Achieved
Construction	Fully Achieved
Transition	Largely Achieved
Management	Fully Achieved

His ratings are Fully Achieved for exploration, construction and management aspects and Largely Achieved for transition aspect. He found all the findings we presented meaningful for their project and true. He mentioned that collecting project specific historical estimation and actual data for more reliable estimations and establishing dependency relations between design components for risk evaluation are new concepts to him which are also valid and required for the sake of project.

He explained the reason of "largely achieved" rating for transition aspect as follows:

He called attention of the need of obtaining feedback from the customers and end users after the delivery of software product. He mentioned that even if the software development processes are highly matured, there might be issues related to the hardware systems and the network and the end product may be perceived as of poor quality because of these reasons. After experiencing a few significant failures, they had decided to monitor the released software by log operations and interfered on time to these kind of failures. We limited the boundary of this thesis with the delivery of the

software product. In the upcoming versions of AgilityMod we might focus on achieving agility and obtaining rapid feedback after the delivery of software product.

Case 3: Organization L, Project #3: To remind, project's aspects agility level are Level 2: Lean, Level 2: Lean, Level 1: Ad-Hoc and Level 3: Effective for Exploration, Construction, Transition and Management aspects respectively.

We discussed the findings with the software product development manager, the scrum master, the director, and the software quality assurance manager. We could obtain ratings of two people: the scrum master and the quality assurance manager. However, the comments of all attendees are given below.

Ratings of the scrum master and the quality assurance manager for the success of the assessment results:

Table 13 Ratings of the Findings-Case 3

Aspects	**Scrum Master**	**Quality Assurance Manager**
Exploration	Largely Achieved	Largely Achieved
Construction	Largely Achieved	Fully Achieved
Transition	Largely Achieved	Fully Achieved
Management	Fully Achieved	Fully Achieved

The scrum master thinks that our findings and suggestions largely overlaps with the current situation. She mentioned that the following findings are false negative. They have already started to construct a dependency analysis among user requirements, they define "epics" and they don't need to assess properness of their requirements to INVEST criteria (Independent, Value-Added, Small, Estimable, Testable) since their requirements already complies with these criteria.

Her low ratio does not arise because of a missing finding but because of false negative findings. On the other hand, as a scrum master she probably thinks that the current process structure is better than currently is. Because the software quality assurance manager (SQAM) mentioned that the traceability issue among software requirements needed to be specified in the presentation. Current relations stored in Jira tool is not at a sufficient level. That is the exact opposite of the ideas of scrum master. He thinks that the positive findings we presented are false positive. Mock-up screens and detailed specifications are not developed, Non-functional requirements are transformed into user stories, Coding standards are applied to improve the code quality, Lessons learned are kept in a wiki based web page and regularly updated, even if we specified they are performed. He also specified that he would prefer the suggested leveling approach for agility improvement but he needs to analyze the results better.

These comments reveal the need for evaluating aspects deeper by collecting and observing more evidences.

The software product development manager who conducts PhD on agile software development topic, mentioned that the presentation covered all the gaps related to their agile processes. He mentioned that he had also previously noticed the problems on developers' testing approach, code integration, the need of developers' and testers' work in a collaborative environment, and doing more investment on code refactoring and

automated tests. He thinks that given suggestions can be applied in a mixed order, the leveling does not make much sense.

The director of software development division thinks that the presentation covers all the problematic areas, there is nothing new to him and the agility levels are meaningful and can be preferred for agility improvement.

***Case 4: Organization I, Project #4*:** To remind, project's aspects agility level are Level 1: Ad-Hoc, Level 2: Lean, Level 1: Ad-Hoc and Level 0: Not Implemented for Exploration, Construction, Transition and Management aspects respectively.

We presented the assessment findings to three people from Project #4, the product owner, software team leader and test engineer. They specified that they did not notice the following issues before our gap analysis and presentation: improving agility awareness of the project team through agile trainings, establishing dependencies among requirements, establishing a measurement and monitoring infrastructure, managing risks, prioritization of backlog items and estimation of requirements items. They found the improvement suggestions meaningful however they mentioned that their priority is to release the software product #4 as soon as possible rather than initiating change on current processes.

They gave the following ratings to specify the overlapping ratio of our findings and their findings. Each person rated his/her own area.

Table 14 Ratings of the Findings-Case 4

Aspects	Product Owner	Software Team Leader	Test Engineer
Exploration	Fully Achieved	not rated by him	not rated by him
Construction	not rated by her	Fully Achieved	not rated by him
Transition	not rated by her	Fully Achieved	Fully Achieved
Management	Largely Achieved	not rated by him	not rated by him

The reason of the lower ratio in management aspect given by product owner is the false positive identification of the two practices which are related to the elaboration of vision and scope.

***Case 5: Organization C, Project #5**:* To remind, project's aspects agility level are Level 3: Effective for Exploration, Construction and Management aspects and; Level 2: Lean for the Transition aspect.

We discussed the findings with program manager, configuration manager and quality assurance team leader of project #5. They gave the following ratings to specify the overlapping scale of our findings and their findings.

Table 15 Ratings of the Findings-Case 5

Aspects	Software Team Leader	Configuration Manager	Quality Assurance Manager
Exploration	Fully Achieved	Fully Achieved	Fully Achieved
Construction	Fully Achieved	Fully Achieved	Fully Achieved
Transition	Fully Achieved	Fully Achieved	Fully Achieved
Management	Fully Achieved	Fully Achieved	Fully Achieved

These people have been working on establishing an agile software development for 15 months and they know what their next improvements are. They mentioned that the results are fully compatible with their own findings. The needs we specified related to the technical dept management and external agile adoption trainings are new concepts to them.

Case 6: Organization NT, Project #6: To remind, Project's aspects agility level are specified as "Level 2: Lean" for all the four aspects.

We discussed the findings with software verification manager, configuration manager, technical project manager and quality assurance specialist. They gave the following ratings to specify the overlapping scale of our findings and their findings.

Table 16 Ratings of the Findings-Case 6

Aspects	**Software verification manager**	**Configuration manager**	**Technical project manager**	**Quality assurance specialist**
Exploration	Fully Achieved	not rated by him	Fully Achieved	Fully Achieved
Construction	Fully Achieved	not rated by him	Fully Achieved	Fully Achieved
Transition	Largely Achieved	Fully Achieved	Fully Achieved	Fully Achieved
Management	Fully Achieved	not rated by him	Fully Achieved	Fully Achieved

Downstream teams who are in contact with the customers, responsible for developing customer support documentation or "exe" files for software installation or providing direct support to customers, work based on the principles of waterfall approach. Software verification manager thinks that there must be a synergy and interaction between development teams and the downstream team. Deliverables are highly overlap between R&D team and downstream teams. He thinks that AgilityMod needs to question the practices after the deployment process since these are highly correlated with software quality.

Performing quality audits, managing technical dept, assessing agile alignment among team members are new concepts to representatives. They mentioned that they are currently working on adopting agile engineering practices to their processes and apply our improvement suggestions based on an improvement plan.

4.2.4. Discussion

As described in the previous sub-sections, multiple case study was applied as the research methodology to evaluate AgilityMod.

We applied the model in six different organizations, in six projects from different domains. Demographics information about the cases are provided in Table 17 covering the domain of the organization and domain of the assessed project, project team size, team location, utilized programming language, customer location with respect to the development team, customer communication ways, preferred management approach, length of each iteration and consistency in maintaining iteration length, developed

Table 17 Demographics of the Cases

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
	Org NM, P. #1	**Org G, P. #2**	**Org L, P. #3**	**Org I, P. #4**	**Org C, P. #5**	**Org NT, P #6**
Domain of the Organization	**Tech-media company**	**Government IT organization**	**ERP solutions company**	**Various communication systems**	**Software and Internet Security**	**Cloud and multimedia solutions**
Project Team Size	22 full-time	23 full-time	19 full-time	7 full-time, 3 part-time	22 full time	45 full time
Domain of the assessed project based on CHAR method	Controlling Information System	Controlling Data System	Non-Specific (Complex) System	Information System	Complex Data-Driven Control System	Complex Control System
Team Location	Local	Local	Local	Local	Local	Local
Programming Language	PHP	J2EE, Flex	Pascal	PHP	PHP, Java Phyton, Cassandra	Java, C++, C, Java Script, HTML5
Customer Location	Internal development, no direct customer	External, in another location	Both Internal and External	Internal development, no direct customer, COTS product	External, in another country	External, in another location
Customer Communication Ways	Face to face	e-mail, phone, face to face	Email, phone over partners, over support portal, face to face	Face to face	Tele- conference	Tele- Conference, e-mail
Preferred Management Approach	Non-specific	Scrum	Adjusted Scrum	Scrum	Scrum	Scrum
Iteration Cycle Time and consistency	7 to 10 days consistent	30 days consistent	45 days Consistent (7 weeks)	15 days to 30 days non consistent	3 weeks, consistent	3 weeks, non-consistent

Source LOC	Not Available	7 millions	6 millions	630.000	Not Available	500.000
Approx. % of Code Coverage with Unit Tests	Not Exists	Changes based on modules, 54,5% on average	Code coverage over automated tests %23	Not Exists	Approximately %10	Not exists
Continuous Integration	Not Exists	Applied	Not Exits	Not Exists	Initiated	Initiated
Type of the Agreement with the Customer	Internal development no contract.	Contract Based	Internal development no contract.	Internal development no contract.	Internal development	Contract based

source of lines of code, approximate percentage of code coverage with unit tests, appliance of continuous integration and the type of the agreement signed with customer.

We applied the model in various domains ranging from technical media, home appliances, ERP solutions, multimedia solutions and e-governance solutions. The team sizes of the assessed projects change between 6 employees to 45 employees. In projects #1 and #4, there is no external customer, teams decide and analyze product specifications themselves. In projects Delta, #2, #4 and #6 external customers who are working in different locations provide specifications to the teams through product owners. In project #3 both internal and external customers specify the requirements. For the customers who locate in different offices, project teams establish various solutions to improve communication such as frequent teleconferencing, and customer-side face to face meetings and maintaining communication matrixes for the monitoring the efficiency of the communication.

Project teams deliver working software in different iteration lengths. The shortest iteration length is in Project #1 with 7 days and the longest iteration length belongs to project #3 with 45 days. The length of the iteration is an indicator how fast the feedback is obtained and as well as the agility. In all nine case studies, we evaluated the projects' processes from the perspective of four aspects: exploration, construction, transition and management. In terms of these four aspects, overall results of the cases studies are listed in Table 18.

Table 18 Overview of multiple case study results

Case No	Org.	Project Name	Exploration Aspect Level	Construction Aspect Level	Transition Aspect Level	Management Aspect Level
Case 0	Exploratory case study		L1: Ad-Hoc	L0: Not Implemented	L1: Ad-Hoc	L0: Not Implemented
Case 1	NM	#1	L1: Ad-Hoc	L1: Ad-Hoc	L0: Not Implemented	L0: Not Implemented
Case 2	G	#2	L3: Effective	L3: Effective	L3: Effective	L3: Effective
Case 3	L	#3	L2: Lean	L2: Lean	L1: Ad-Hoc	L3: Effective
Case 4	I	#4	L1: Ad-Hoc	L2: Lean	L1: Ad-Hoc	L0: Not Implemented
Case 5	C	#5	L3: Effective	L3: Effective	L2: Lean	L3: Effective
Case 6	NT	#6	L2: Lean	L2: Lean	L2: Lean	L2: Lean

We draw the graph below to better present the distribution of the achieved levels for the each aspect of each case.

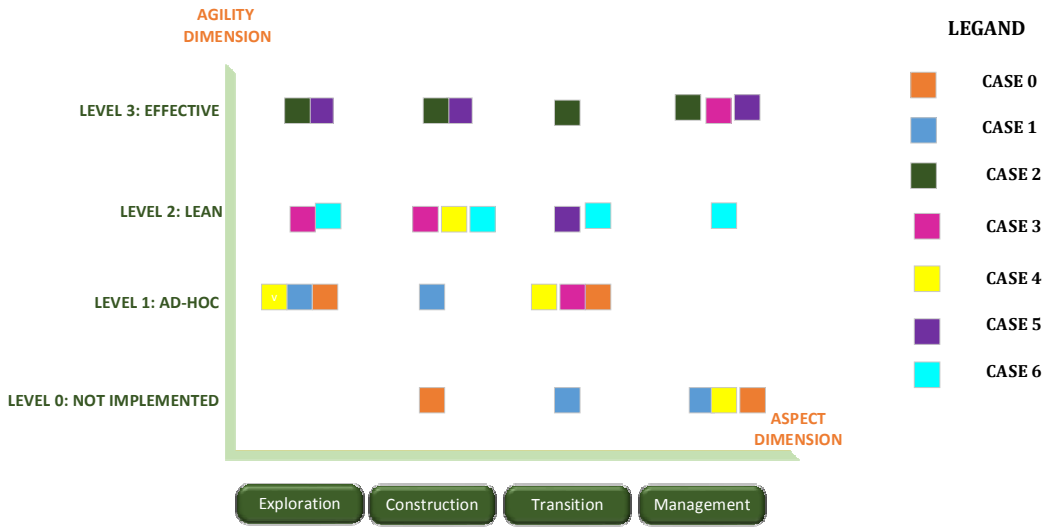

Figure 22 Distribution of achieved agility levels

Considering the results of case studies with respect to the research questions, we achieved the following results.

RQ1: How suitable is the Software Agility Assessment Reference Model (AgilityMod_v3.0) to be used with the purpose of identifying aspects' agility, identifying agility gaps and providing roadmaps for improving agility in software projects?

Considering the multiple case study results, the opinions of the interviewees on the results discussed in section 4.2.3 and the feedbacks of experts, we conclude that we could use AgilityMod to identify the agility gaps in projects, to specify agility levels of aspects and to provide roadmaps to organizations for agility improvement.

As mentioned before in section 3.2.2 AgilityMod_v2.0 has been reviewed by three experts. They gave comments directly on the Model and filled in a questionnaire which aims to specify the Model's achievement degree of six criteria: fitness for purpose, completeness, definition of agile levels, objectivity, correctness and consistency.

In the multiple case study conducted, we observed the occurrence of all the three agility levels from L1 to L3 for Exploration and Construction aspects. However, we couldn't observe the occurrence of L1 for Management aspect and L0 for Exploration aspect. Observation of every agility level for each aspect shows both the broad perspective of the case studies conducted and the capability of AgilityMod in specifying and representing diversities between agility levels.

Two of the experts (Expert A and Expert C) mentioned that the component descriptions are clear enough to perform agility assessment and the model is capable of providing directions for improvement on agility and can be used as a roadmap by organizations for getting better at agility. Expert A expressed his ideas in these topics as follows:

"*The model aims to bring a maturity view on the agile principles, and I believe it is a successful model. Using ISO 15504 as a reference model supports the validity of the model*

and increases the possibility of usage among organizations. The model perfectly fits the need of providing roadmap by organizations for getting better at agility"

The capability of AgilityMod in identifying agility gaps was evaluated in the interviews conducted with assessment team members after presenting the assessment results to them. We had chance to discuss the assessment findings for each case study. The feedbacks obtained for case studies are in the previous section in detail (section 4.2.3). We gathered all the ratings obtained from aspect owners in Table 19 below. In order to construct the table we obtained the median of the ratings if the assessment findings were rated more than one person. In the overall, 87.5 % percent of the evaluation indicates that the findings and improvement suggestions fully overlap with current problem in the projects. The remaining 12.5 % thinks that aspect findings largely overlaps with current problems. Achieving such high ratios for finding the gaps in the projects is an indicator of how successful the Model in revealing agility improvement opportunities and the potential of the Model for the usage of agility assessment. On the other hand, we are aware of the need to develop an assessment approach.

Table 19 Ratings of the Findings Based on the Cases

Aspects	CASE 1	CASE 2	CASE 3	CASE 4	CASE 5	CASE 6
Exploration	FA	FA	LA	FA	FA	FA
Construction	FA	FA	FA	FA	FA	FA
Transition	FA	LA	FA	FA	FA	FA
Management	FA	FA	FA	LA	FA	FA

In order to evaluate the efficiency of agility levels, we asked people who were involved in the assessment process if they would prefer to apply improvement suggestions for each agility stage, in the order we specified. The answers we obtained are varying. Interviewed people were agree on applying the improvement suggestions in the order we specified for Case 1, Case 2, Case 4, Case 5 and Case 6 whereas for Case 3 the most of the interviewees mentioned that they would not need an order for implementing the suggestions, since they think the suggestions are independent from each other. It is true that there is an independency in improving aspects, however, there is integrity within aspects which requires being lean first and then effective. Giving answer to that question requires deeper analysis of the assessment reports and answering might just after the findings presentation might be misleading.

RQ2: What are the strengths and weaknesses of AgilityMod?

We interpreted the strengths and weaknesses of AgilityMod based on the feedbacks of experts and the results of the multiple case study.

In the Model, we described the agility in an abstract way to cover various agile methods and approaches. Therefore it is very important the Model components' and component descriptions' both cover all agile principles in an abstract way and be independent of any agile method. Experts evaluated the Model from these perspectives and rated as fully or largely achieved. Expert A found the level of abstraction appropriate when the audience of the model is considered as daily agile practitioners. The more you keep the abstraction

at a reasonable level, the more the experience and knowledge of the assessor becomes important. The target group that is expected to use AgilityMod for assessment are experts who have specific knowledge and experience in the agile domain.

Expert C gave specified the descriptions of components that is too specific or valid for a particular agile method. The Model is updated considering the expert comments and 3rd version of it is published.

In terms of "consistency", experts commented directly on the Model to specify minor inconsistencies and concluded that the Model is internally consistent and does not include any logical conflicts.

All experts think that the Model is "correct" such that all component descriptions are compatible with agile values and principles.

One of the requirements of an assessment model is to achieve a required level of "objectivity" in order to guarantee the repeatability of the assessment results. AgilityMod aims to achieve "the objectivity" through clear description of aspect purpose and outcomes, and aspect and agility practices. AgilityMod uses the common rating scale with ISO/IEC 15504 [79] that clearly specifies the ranges for rating. In terms of objectivity Experts A mentions that clarifying the normative and informative features of the Model would increase the objectivity. As in ISO/IEC 15504 all model elements except for the aspect purpose and outcomes are informative. Expert C calls attention to the need for a rating scheme for assessing multiple agile projects and specifying agility of an organization rather than project basis. We are going to define the rules for assessing agility of organizations, however, this improvement is not in the scope of this thesis. Therefore we consider that these comments of experts do not violate "objectivity" characteristic of AgilityMod.

In addition to the comments of experts, we also had very specific observations related to the Model while conducting the multiple case study.

At the beginning of this study we have the idea based on our past experiences that successful organizations that are characterized with quality software are not just applying agile practices, rather they are adopting their systems conforming to the agile principles. From this perspective, one of the most important characteristics of AgilityMod is that we do not only evaluate the existence of some specific agile practices such as performing daily stand-up meetings or pair programming or collective code ownership instead we evaluate the aspects from a holistic approach and try to understand for example if the teams are capable of keeping the design structure sound while responding to the changes quickly. In order to achieve this purpose we defined specific aspect practices such as "exploring the design through technical search activities and regular dependency analysis" or changing code with control in addition to collective code ownership and many more.

Even if the model is "complete" and "objective" it is suggested to conclude about the ratings of each practice with the consensus of assessment team including a lead assessor and assessment team members. The Part 3 of ISO/IEC 15504 standard provides an assessment approach at a high level including planning, data collection, validation and reporting phases and roles and responsibilities for an assessment. What we are looking for in terms of an assessment approach is a more complete description of different types

of appraisals, the evidences to be gathered, the coverage criteria of organizational units and appraisal team requirements. A well-defined approach including these aspects can be found in Standard CMMI Appraisal Method for Process Improvement (SCAMPI). However, in the scope of this thesis we did not aim to provide an appraisal approach and recorded this opportunity as a future study.

When we criticize the Model in terms of its components and components descriptions, the following issues emerge:

One of the attributes of the third level is the "Learning attribute". In the scope of the "Learning" attribute we assess how the aspects serve for the purpose of organizational learning and improvement. We aim to capture the evidences for collaborative work and shared responsibility in the conduct of aspect practices, agile leadership styles and encouraging people in the organization to participate in learning, teaching and improvement. What we observe in the multiple case study is if collaborative work or self-organization is established in the project, it is not just valid for one specific aspect but valid for the other aspects as well. Therefore, when an evidence is observed during the assessment of an aspect through an agility practice, there is no need to question the same agility practice for the remaining aspects. Because the answer is pretty much the same. In some cases there might be significant differences in the answers of the fourth agility practice of the learning attribute which is "collecting measures to support learning and improvement". We mention this situation as an issue to be considered because the improvement suggestions repeat itself in the assessment reports.

It was observed that the positive and negative evidences for agility attributes of second level may also be the evidences for some of the agility practices at the third level. For example, to employ minimally sufficient ceremony in any kind of activity we expect aspects' be evaluated regularly to eliminate redundancies and non-value added activities. On the other hand the purpose of learning attribute at the third level is to learn from past experiences and improve continuously which also requires performing regular retrospective studies. What we need to mention here is that the components at the second and the third level may trigger each other and there may not be clear cut distinctions between the components.

Another conflict was detected in the Exploration aspect practice "E.AP3: Detect and resolve conflicts of requirements artifacts" and Iterative attribute practice "GP 2.1.2 Communicate effectively", since detecting and resolving conflicts in requirements artifacts is highly correlated with effective communication. To resolve this issue one option is to remove the third practice of exploration aspect (E.AP3), however since it is one the major problems in exploration activities we preferred to keep it as is to call attention to this problematic area in software development.

Another conflict among aspect and agility practices was detected between the "Simple" attribute practice "GP 2.2.1: Balance the predictive work and adaptive work" and "T.AP5: Make the progress visible" from Transition aspect and "E.AP6- Make the artifacts visible to everyone" from Exploration aspect. In order to resolve the conflict we removed the emphasis "Work items in the workflow are visualized" specified in GP 2.2.1.

On the other hand, we also confront the conflicting practices issue in ISO/IEC 15504 such that when "PA1.1: Process Performance" attribute is achieved, "PA 2.1: Performance

Management" attribute is partially achieved as well. Because it is very difficult to separate process performance and performance management totally.

As discussed earlier in subchapters 3.2.2 and 4.1.3.2, we removed the "Culture" aspect from aspect dimension since its practices had in a significant conflict with "Learning" attribute practices and "Management" aspect practices and extended other attribute or aspect practices to cover the unique practices of culture aspect. In the conduct of the multiple case study we also checked if any of removed culture aspect practices are skipped or not. We are sure that none of the culture practices are skipped. However, we also observed that it was a better idea to present the results in terms of a dedicated aspect for culture rather than specific practices since the organizational culture has a significant impact on agile transformation and it is easier to call attention of team members by making the cultural behaviors clearly visible.

In the conduct of multiple case study, we assessed the projects' agility by meeting one to four team member from different roles. The Model includes specific practices related to culture of project teams and communication interfaces and effectiveness (see GP 2.1.2, 3.2.1 and 3.2.2). We directed the questions related to these generic agility practices to only the interviewed people. When an assessment approach is defined, it is suggested to include whole project team to the assessment of such practices to capture the differences in perceptions of people and reach more accurate results.

4.2.5. Validity Threats

Because of its nature, case study approach use quantitative data and provide solutions in its own context. It is possible to arise some validity concerns in case study research. [98]. Below we discuss the limitations of the multiple case study in terms of construct, internal and external validity.

Construct Validity

This type of validity considers if the constructs in the case study are well-structured or subjective to the judgment [98]. In other words, construct validity concerns if what being looked for in a study is exactly specified or not.

There is a threat on interpretation of the constructs discussed in the interview questions and questionnaire by the researcher and the interviewed people in the same way [99]. In order to prevent this threat we explained each question by providing examples.

There is a thread on the selection of the cases for the multiple case study. In order to prevent the thread we selected cases based on pre, informal discussions about the application of agile practices in the candidate projects and applied the case in 6 projects to observe all agility instances.

Internal Validity

Internal validity also known as logical validity deals with the relationships between variables and results and concerns if the researcher is aware of all the factors affecting his/her study [100].

In order to eliminate any bias in assessment findings we discussed the findings with aspect owners in project teams.

External Validity

External validity concerns the generalizability of the case study results and evaluates if the study is valid in its own setting or applicable in other settings as well [98, 99].

We performed the multiple case study in different business domains including a government organization, a technical-media company, an ERP solutions company, a home appliances company, a software security and a cloud and multimedia solutions company as indicated in Table 17. The Model is applied in a controlling information system, a controlling data systems, an information system, a non-specific complex system, a complex data-driven control system, and a complex control system projects. We did not observe any difference or difficulty for the application of Model in these business and project domains. Therefore, we conclude that the Model can be applied being independent of any business and project domains.

By considering the results of the multiple case study (section 4.2) and the exploratory case study (section 4.1), we could observe three agility levels (L1, L2, L3) for Exploration aspect, all four levels (L0, L1, L2, L3) for Construction and Transition aspects, three agility levels (L0, L2, L3) for Management aspect. The distribution of agility levels for each aspect has shown in Figure 22 in section 4.2.4. For the agility levels that couldn't be observed, more case study are needed to be planned. Observing an agility level for an aspect means that the defined levels of aspects are valid.

CHAPTER 5

CONCLUSION

Agile software development methods are more than welcomed by the software community in recent years. However, there have been issues related to the adoption of agile values and principles and transformation of organizations. There might be various reasons of this. Misinterpretation of agile principles and values and adopting partial solutions with few agile practices instead of holistic approaches prevented organizations obtain full benefits from tempting agile methods.

In this thesis, we propose an Agility Assessment Reference Model, AgilityMod, to be utilized for the appraisal of software projects from agility perspective. In this chapter, the summary of the thesis study and contributions achieved by the proposed Agility Assessment Reference Model are presented. The suggestions for future work are given.

5.1. Summary of the Thesis Study and Contributions

Before proposing the Model, we performed a comprehensive literature review study to identify the models developed with similar purposes. Following that, we conducted a case study with the current agile maturity/assessment models to evaluate their quality and capability of agility assessment. The case study included application of five models in a software development organization and evaluation of the models. The models were evaluated based on six criteria: fitness for purpose, definition of agile level, completeness, correctness, consistency and objectivity. The results of the case study indicated that none of the models fully achieve the specified criteria. We published this study in one of the remarkable conferences in software process improvement area, Software Process Improvement and Capability Determination (SPICE) conference in 2013 [39].

After that, we developed the first version of AgilityMod [95]. AgilityMod_v1.0 included five aspects: Exploration, Construction, Transition, Management and Culture and four agility levels: Not Implemented, Ad-Hoc, Lean and Effective. We performed an exploratory case study in order to observe the applicability of the Model and determine improvement opportunities for the Model. We chose the same case that we used in our previous study to better specify the assessment capability of our model. The projects are found at Level 1: Ad-Hoc, Level 0: Not Implemented, Level 1: Ad-Hoc, Level 0:Not Implemented and Level 0:Not Implemented for Exploration, Construction, Transition, Management and Culture aspects respectively. During the case study we specified some redundancies in practices, missing practices or excess usage of practices. These were

discussed at section 4.1.3.2 in detail. We also published this study in SPICE conference in 2014 [93].

We updated the Model based on the exploratory case study findings, and published AgilityMod_v2.0 as a technical report [94]. The major difference of second version from the first version is that we removed the conflicting practices and conflicting aspect: "Culture" from the Model. We observed that cultural elements and practices are covered under "Learning" attribute and "Management" aspect. For the practices of the Culture aspect that had not been previously represented, we ensured that they are included in other components of the Model.

The second version of AgilityMod was reviewed by three experts who have expertise in software process improvement and agile software development domains. They commented on the Model and we discussed their comments through teleconference meetings one by one. We also asked them to answer a set of questions that assesses the Model based on six criteria that we previously set in [39]. They provided ratings for each of the criteria. According to that, AgilityModv2.0 largely achieves fitness for purpose, completeness and consistency criteria, fully achieves correctness criterion and partially achieves definition of agile levels and objectivity criteria. We improved the Model based on their feedback. Accordingly, in the 3rd version, the model was extended to cover measures, and purified by removing the definitions and terms specific to Scrum, Kanban or XP. A detailed explanation of this update process was discussed in section 3.2.2.

Following this update we planned and conducted a multiple case study including six cases from different business and technical domains. After conduct of each case study, we prepared an assessment report including the situation of the project and our improvement suggestions for the findings. In order to validate the assessment results, we discussed the findings with aspect owners, who had been involved in the assessment process and also high level managers in some cases. For 6 cases, 24 aspects' agility level are evaluated and reported in total. Details of this case study was discussed in section 4.2. The multiple case study showed that AgilityMod_v3.0 can be applied to identify the agility gaps in projects, to specify agility levels of aspects and to provide roadmaps to organizations for agility improvement.

As described above, we preferred an iterative and incremental approach for the development of the Model from the beginning of the study. We developed it, applied it in the industry and improved recursively.

The major contribution achieved in this study is the Agility Assessment Reference Model, designed to be a complete solution for agility assessment with its fully compatible structure with agile values and principles. In this domain, it is not possible to find a complete solution both covering agile principles and values and agile practices.

We could assess "Agility" of a project in terms of four aspects instead of checking compatibility to some agile practices. In AgilityMod, we do not only evaluate the existence of some specific agile practices such as performing daily stand-up meetings or pair programming or collective code ownership, instead we evaluate the aspects from a holistic approach and understand for example, if the teams are capable of keeping the design structure sound while responding to the changes quickly.

In this study, we observed applicability of the Model through multiple case studies, none of the models in the literature include such applications.

The Model provides guidance to assessor with specific and generic practices, example work products and agile elaborations.

The Model has been developed based on the meta-model of ISO/IEC 15504. Although AgilityMod uses the structure of ISO/IEC 15504 and shares commonalities, we needed to change its components in order to achieve compatibility with agile process architecture. AgilityMod defines dimensions, aspects (instead of processes in 15504), aspect attributes (instead of process attributes), aspect practices (instead of base practices) and generic agility practices (instead of generic practices). With this structural changes, the Model gained a characteristic specific to the agile domain. The benefit obtained by choosing ISO/IEC 15504 for the meta-model is the familiarity of people to ISO/IEC 15504 in software process improvement domain and adaptation potential AgilityMod with ease.

We defined two new dimensions in the Model: agility dimension and aspect dimension. These two dimensions allow us to specify the agility in terms of aspects. The leveling approach of AgilityMod is a type of continuous representation [101] that uses agility levels to characterize the state of projects' aspects. While we are designing AgilityMod in a continuous mode, the thing in our minds was to provide enough flexibility to organizations to focus them on improvement of different aspects for a specified time interval and observe the improvements for each aspect separately. Each aspect has its own practices describing the agile elaborations, example work products and resources. Normative elements of each aspect are aspects' purpose and outcomes. The purpose and outcomes of an aspect may be achieved by using different types of agility practices and evidences. In addition, we defined "fallacies" for each aspect that needs to be considered as negative evidences during assessment and kind of warning signs for the assessors. Developing the Model based on meta-structure of ISO/IEC 15504 brings the possibility of transforming the Model into an ISO standard.

Providing and obtaining fast feedback, achieving technical excellence, communicating effectively, developing software iteratively and incrementally, balancing the flow of work and work products in the process are some of the key elements in agile software development which are published as twelve agile principles [31]. The agility dimension of AgilityMod reflects and embraces these twelve agile principles.

Although we obtained very positive feedbacks from representatives, they may make emphasis on conflicting issues. While the scrum master find our judgment very strict, the quality assurance and test manager from the same project may find our suggestions less when their problem are considered. These arguments revealed that there is a need to develop an assessment approach and apply AgilityMod through this systematic approach. We evaluated one project for each of the organization, however, more projects needs to be evaluated both to improve the reliability of the results and to make inferences throughout organizations.

An assessment approach needs to specify the number of the projects to be assessed based on the characteristics of organization, the complete description of different types of appraisals, the evidences to be gathered, the coverage criteria of organizational units and appraisal team requirements.

Finally, the multiple case study results showed that the Model is successful at identifying agility gaps at different levels of agility and capable of proving solutions for high agility level project. Compared to the other models in the literature, AgilityMod defines practices, provides a holistic agility assessment approach not just covering agile practices but also thinking software development as a whole.

5.2. **Future Work**

We identified the following improvement opportunities regarding AgilityMod:

- Development of a self-agility-assessment approach covering a comprehensive set of questions and alternative answers that are compatible with AgilityMod. Publish of the approach over internet and collection of new assessment data from various software organizations from different countries and benchmarking the data.
- Maintaining the interaction with ISO community to transform AgilityMod into an ISO standard.
- Development of an assessment approach defining the characteristics of assessment teams, roles and responsibilities, and the rules for selection of organizational units for the generalization of agility results in organizations.
- Development of an agility assessment tool regarding AgiltyMod.
- Performing new case studies where assessment teams are involved.

REFERENCES

[1] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*: IBM Press, 2012.

[2] K. Schwaber, *Agile project management with Scrum* vol. 7: Microsoft press Redmond, 2004.

[3] A. Sidky, "A structured approach to adopting agile practices: The agile adoption framework," Virginia Polytechnic Institute and State University, 2007.

[4] A. Elssamadisy, *Agile adoption patterns: a roadmap to organizational success*: Addison-Wesley Professional, 2008.

[5] P. F. Drucker, "Management's new paradigms," *Forbes Magazine,* vol. 10, p. 98, 1998.

[6] T. Dingsøyr, T. Dybå, N. Brede Moe, T. Dingsøyr, and T. Dybå, "Agile Software Development," *Agile Software Development: Current Research and Future Directions, ISBN 978-3-642-12574-4. Springer-Verlag Berlin Heidelberg, 2010,* vol. 1, 2010.

[7] H. Merisalo-Rantanen, T. Tuunanen, and M. Rossi, "Is extreme programming just old wine in new bottles: A comparison of two cases," *Journal of Database Management (JDM),* vol. 16, pp. 41-61, 2005.

[8] R. Baskerville, L. Levine, J. Pries-Heje, B. Ramesh, and S. Slaughter, "Balancing quality and agility in Internet speed software development," in *23rd International Conference on Information Systems*, Barcelona, Spain, 2002.

[9] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New directions on agile methods: a comparative analysis," in *25th International Conference on Software Engineering*, 2003, pp. 244-254.

[10] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations," *IEEE Software,* vol. 22, pp. 30-39, 2005.

[11] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*: Addison-Wesley Professional, 2012.

[12] A. Cockburn, *Agile software development: the cooperative game (agile software development series)*: Addison-Wesley Professional, 2006.

[13] J. Highsmith, "What Is Agile Software Development?," *The Journal of Defense Software Engineering,* vol. 15, pp. 4-9, 2002.

[14] J. Stapleton, *DSDM Dynamic Systems Development Method: the method in practice*: Cambridge University Press, 1997.

[15] K. Schwaber, "Scrum development process," in *Business Object Design and Implementation*, ed: Springer, 1997, pp. 117-134.

[16] M. Aoyama, "Agile software process model," in *Computer Software and Applications Conference, 1997. COMPSAC'97. Proceedings., The Twenty-First Annual International*, 1997, pp. 454-459.

[17] A. Cockburn, *Crystal clear: a human-powered methodology for small teams*: Addison-Wesley Professional, 2004.

[18] A. Cockburn, *Surviving object-oriented projects: a manager's guide*: Addison-Wesley Longman Publishing Co., Inc., 1998.

[19] A. Cockburn, "Writing effective use cases, The crystal collection for software professionals," ed: Addison-Wesley Professional Reading, 2000.
[20] K. Beck, *Extreme programming explained: embrace change*: Addison-Wesley Professional, 2000.
[21] K. Beck, "Embracing change with extreme programming," *Computer,* vol. 32, pp. 70-77, 1999.
[22] M. A. Cusumano and D. B. Yoffie, "Software development on Internet time," *Computer,* vol. 32, pp. 60-69, 1999.
[23] R. Baskerville, L. Levine, J. Pries-Heje, B. Ramesh, and S. Slaughter, "How Internet software companies negotiate quality," *Computer,* vol. 34, pp. 51-57, 2001.
[24] R. Baskerville and J. Pries-Heje, "Racing the E-bomb: How the Internet is redefining information systems development methodology," in *Realigning research and practice in information systems development*, ed: Springer, 2001, pp. 49-68.
[25] J. A. Highsmith and K. Orr, *Adaptive software development: a collaborative approach to managing complex systems*: Dorset House Pub., 2000.
[26] A. Hunt, *The pragmatic programmer: from journeyman to master*: Addison-Wesley Professional, 2000.
[27] S. R. Palmer and M. Felsing, *A practical guide to feature-driven development*: Pearson Education, 2001.
[28] S. W. Ambler, *Agile modeling*: Wiley, 2002.
[29] M. Poppendieck and T. Poppendieck, *Lean software development: An agile toolkit*: Addison-Wesley Professional, 2003.
[30] K. Beck, *Test-driven development: by example*: Addison-Wesley Professional, 2003.
[31] (2001). *Agile Manifesto*. Available: www.agilemanifesto.org
[32] A. Tomasini and M. Kearns, *Agile Transition: What you need to know before starting*: InfoQueue Enterprise Software Development Series, 2012.
[33] J. Highsmith, *Agile project management: creating innovative products*: Pearson Education, 2009.
[34] A. Sidky and G. Smith, *Becoming Agile in an imperfect World*: Manning Publications Co., Greenwich CT, 2009.
[35] VersionOne, "8th Annual State of Agile," http://stateofagile.com/8th-annual-state-of-agile-form/2013.
[36] S. Ambler. (2013). *IT Project Success Rates Survey Results*. Available: http://www.ambysoft.com/surveys/success2013.html
[37] T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, and M. Biro, "Agile Maturity Model: A Synopsis as a First Step to Synthesis," in *Systems, Software and Services Process Improvement*, ed: Springer, 2013, pp. 214-227.
[38] T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, and M. Biro, "Agile maturity model: analysing agile maturity characteristics from the SPICE perspective," *Journal Of Software: Evolution And Process,* 2013.
[39] Ö. Özcan Top and O. Demirörs, "Assessment of Agile Maturity Models: A Multiple Case Study," in *Software Process Improvement and Capability Determination*, Bremen, Germany, 2013, pp. 130-141.
[40] C. Patel and M. Ramachandran, "Agile Maturity Model (AMM): A Software Process Improvement framework for Agile Software Development Practices," *International Journal of Software Engineering,* vol. 2, pp. 3-28, 2009.
[41] A. Yin, S. Figueiredo, and M. Mira da Silva, "Scrum Maturity Model: Validation for IT organizations' roadmap to develop software centered on the client role," in

ICSEA 2011, The Sixth International Conference on Software Engineering Advances, 2011, pp. 20-29.
[42] R. Benefield, "Seven Dimensions of Agile Maturity in the Global Enterprise: A Case Study," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, 2010, pp. 1-7.
[43] J. Erickson, K. Lyytinen, and K. Siau, "Agile modeling, agile software development, and extreme programming: the state of research," *Journal of Database Management (JDM),* vol. 16, pp. 88-100, 2005.
[44] J. Shore and S. Warden, *The art of agile development*: O'Reilly Media, 2007.
[45] J. W. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*: Sage Publications, Inc, 2009.
[46] Q. U. Press, "Qxford English Dictionary," ed. http://www.oed.com/.
[47] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," ed: VTT Finland, 2002.
[48] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Communications of the ACM,* vol. 48, pp. 72-78, 2005.
[49] B. Boehm, "Get ready for agile methods, with care," *Computer,* vol. 35 pp. 64-69, 2002.
[50] K. Conboy, "Agility from first principles: reconstructing the concept of agility in information systems development," *Information Systems Research,* vol. 20, pp. 329-354, 2009.
[51] J. Highsmith, *Agile software development ecosystems*: Addison-Wesley Longman Publishing Co., Inc., 2002.
[52] T. Stober and U. Hansmann, *Agile Software Development: Best Practices for Large Software Development Projects* vol. 3: Springer, 2010.
[53] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," *Computer,* vol. 34, pp. 120-127, 2001.
[54] S. Ambler, *Agile database techniques: Effective strategies for the agile software developer*: John Wiley & Sons, 2012.
[55] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer,* vol. 36, pp. 47-56, 2003.
[56] P. Abrahamsson, N. Oza, and M. T. Siponen, "Agile Software Development Methods: A Comparative Review1," in *Agile Software Development*, ed: Springer, 2010, pp. 31-59.
[57] J. Martin, *Rapid application development*: Macmillan Publishing Co., Inc., 1991.
[58] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*: Prentice-Hall, 2002.
[59] S. Ambler, *Agile modeling: effective practices for extreme programming and the unified process*: John Wiley & Sons, 2002.
[60] P. Kruchten, *The rational unified process: an introduction*: Addison-Wesley Professional, 2004.
[61] S. Bayer and J. Highsmith, "RADical software development," *American Programmer,* vol. 7, pp. 35-35, 1994.
[62] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology,* vol. 50, pp. 833-859, 2008.
[63] J. Humble and R. Russell. (2009, The Agile Maturity Model Applied to Building and Releasing Software
[64] N. Malic, "Simple Life Cycle Agile Maturity Model," ed.

[65] M. Proulx. (2010). *Yet Another Agile Maturity Model (AMM)– The 5 Levels of Maturity*. Available: http://analytical-mind.com/2010/07/12/yet-another-agile-maturity-model-the-5-levels-of-maturity/
[66] S. Jayaraj. (2007). *The Agile Maturity Model*. Available: http://whattodowearelikethatonly.blogspot.com/2008/08/agile-maturity-model.html
[67] S. Ambler. (2010). *The Agile Maturity Model (AMM)*. Available: http://www.drdobbs.com/architecture-and-design/the-agile-maturity-model-amm/224201005
[68] D. J. Anderson, *Agile management for software engineering: Applying the theory of constraints for business results*: Prentice Hall Professional, 2003.
[69] U. Banerjee. (2011). *Agile Maturity Model, Three Different Approaches*. Available: http://setandbma.wordpress.com/2011/11/30/agile-maturity-model/
[70] R. Bavani. (2011). *Distributed Agile: The Maturity Curve*. Available: http://blogs.mindtree.com/distributed-agile-maturity-curve-part-1Available: http://blogs.mindtree.com/distributed-agile-the-maturity-curve-part-2
[71] S. Ronen. *Agile Testing Maturity Model*. Available: http://www.slideshare.net/AgileSparks/atmm-practical-view
[72] D. Woods. (2011). *An Agile BI Maturity Model*. Available: http://www.forbes.com/sites/danwoods/2011/10/26/an-agile-bi-maturity-model/
[73] I. O. f. Standardization and I. E. Commission, "ISO/IEC 15504 Part 7-Information technology -- Process assessment -- Part 7: Assessment of organizational maturity," ed, 2008.
[74] R. Kneuper, *CMMI: Capability Maturity Model Integration A Process Improvement Approach*: Rocky Nook, 2008.
[75] A. Sidky, J. Arthur, and S. Bohner, "A disciplined approach to adopting agile practices: the agile adoption framework," *Innovations in systems and software engineering,* vol. 3, pp. 203-216, 2007.
[76] S. Ambler. (2009). *The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments*. Available: ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF
[77] T. Schweigert, R. Nevalainen, D. Vohwinkel, M. Korsaa, and M. Biro, "Agile maturity model: oxymoron or the next level of understanding," in *Software Process Improvement and Capability Determination*, ed: Springer, 2012, pp. 289-294.
[78] "ISO/IEC 15504-2:2003 Information technology -- Process assessment -- Part 2: Performing an assessment," ed, 2003.
[79] "ISO/IEC 15504-5:2012 Information technology -- Process assessment -- Part 5: An exemplar software life cycle process assessment model," ed, 2012.
[80] I. O. f. Standardization and I. E. Commission, "ISO/IEC 15504-1:2004 Information technology -- Process assessment -- Part 1: Concepts and vocabulary," ed, 2004.
[81] "ISO/IEC 12207:1995/Amd.1:2002, Information technology — Software life cycle processes," ed, 2002.
[82] "ISO/IEC 12207:1995/Amd.2:2004," ed, 2004.
[83] "ISO/IEC 12207:1995/Amd.1:2002, Information technology — Software life cycle processes," ed, 1995.
[84] C. Bianco, "Agile and SPICE Capability levels," in *Software Process Improvement and Capability Determination*, ed: Springer, 2011, pp. 181-185.

[85] G. Lami and F. Falcini, "Is ISO/IEC 15504 Applicable to Agile Methods?," in *Agile Processes in Software Engineering and Extreme Programming*, ed: Springer, 2009, pp. 130-135.
[86] Ö. Özcan Top, "AgilityMod: Software Agility Assessment Reference Model v3.0," Informatics Institute, METU/II-TR-2014-392014.
[87] Ö. Özcan Top, "AgilityMod: Agility Assessment Model v1.0," Informatics Institute, METU/II-TR-2014-37.
[88] Ö. Özcan Top, "AgilityMod: Agility Assessment Model v2.0," Informatics Institute METU/II-TR-2014-38.
[89] P. Middleton and D. Joyce, "Lean Software Management: BBC Worldwide Case Study," *Engineering Management, IEEE Transactions on,* vol. 59, pp. 20-32, 2012.
[90] L. Adkins, *Coaching agile teams: a companion for ScrumMasters, agile coaches, and project managers in transition*: Addison-Wesley Professional, 2010.
[91] L. Williams, G. Brown, A. Meltzer, and N. Nagappan, "Scrum+ engineering practices: Experiences of three microsoft teams," in *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, 2011, pp. 463-471.
[92] G. Benefield, "Rolling out agile in a large enterprise," in *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, 2008, pp. 461-461.
[93] Ö. Özcan Top and O. Demirors, "Assessing Software Agility: An Exploratory Case Study," in *accepted to be published in Software Process Improvement and Capability Determination conference*, Vilnius, 2014.
[94] Ö. Özcan Top, "Agility Assessment Model v2.0," Informatics Institute METU/II-TR-2014-38.
[95] Ö. Özcan Top, "Agility Assessment Model v1.0," Informatics Institute, METU/II-TR-2014-37.
[96] ISO/IEC, "IS 14143-5 Information Technology – Software Measurement - Functional Size Measurement - Part 5: Determination of Functional Domains for Use with Functional Size Measurement," ed, 2004.
[97] Ö. Özcan Top, "AgilityMod: Software Agility Assessment Reference Model v3.0 Application: Case Study Results," Informatics Institute, METU/II-TR-2014-40,2014.
[98] M. Gibbert, W. Ruigrok, and B. Wicki, "What passes as a rigorous case study?," *Strategic management journal,* vol. 29, pp. 1465-1474, 2008.
[99] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering,* vol. 14, pp. 131-164, 2009.
[100] R. K. Yin, *Case study research: Design and methods*: Sage publications, 2014.
[101] C. Institute, "Capability Maturity Model Integrated-Development," ed, 2010.
[102] M. Fowler. (2000). *The New Methodology*. Available: http://martinfowler.com/articles/newMethodology.html
[103] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*: Pearson Education, 2010.

APPENDIX A

AGILITY MOD: SOFTWARE AGILITY ASSESSMENT REFERENCE MODEL

In this section we describe AgilityMod with its all components.

A.1. Agility Dimension

We described the agility at four levels: Not Implemented, Ad-Hoc, Lean and Effective.

A.1.1. Agility Level 0: Not Implemented

A "Not Implemented" aspect means that its practices either are not achieved or partially achieved. "Not Implemented" aspects do not have the benefit and the outcomes that would be obtained when the aspect practices are fully implemented and they are far away from reaching the agile values.

A.1.2. Agility Level 1: Ad Hoc

On the road to be fully agile, the first step is progressing towards adapting agile practices and agile values into software development life cycle activities. Organizations at this level are capable of performing fundamental development processes such as requirements development, design, coding, integration, testing, and deployment consistently. There are transition attempts towards the agility by exploring best fitting agile practices or approaches.

The single aspect attribute of this level ensures the achievement of aspects describing plain software development life cycle activities. Aspect practices are implemented and aspect purposes are achieved; however agile values and principles are not fully incorporated into aspect practices. Agile elaboration of each aspect practice are given to provide a roadmap for agility improvement. It is possible to observe inconsistencies in performing agile practices. Therefore, teams cannot utilize benefits of agile principles and agile values.

This temporal phase might turn to a chronic situation if the organization remains stick to the unbalanced and undisciplined environment where there is no central dominant agile perception.

Teams performing aspects in an "ad hoc" manner may tend to argue that they are doing agile by implementing a few agile practices. They don't have most of the following indicators of agility:

- keeping the design structure sound while responding to the changes quickly
- planning in an adaptive manner while the requirements and requests of the customer are consistently changing
- being flexible as well as disciplined in implementing agile practices
- learning and improving continuously
- allowing people to make mistakes to be adaptive enough
- balancing predictive up-front work and just-in-time work instead of completely being against of up-front work
- including relevant stakeholders to development and decision making processes regularly
- being consistent in delivering working software with regular intervals
- establishing teams including people from different disciplines and creating environments where people can be creative, are self-possessive for iteration goals and self-organizing
- giving value to people in the organization by training them in novel technologies and agile engineering practices and giving enough time to people to internalize and to implement these new practices
- being transparent in sharing information about the progress and problems to find solutions

This level includes a single attribute:

A.1.2.1. Aspect Attribute 1.1 "Performing Aspect Practices"

"Performing Aspect Practices" attribute is a measure of the extent to which purposes and goals of the aspects are achieved by implementing the related practices described in aspect dimension.

Generic Practice for "Performing Aspect Practices" Attribute

GP 1.1.1 Perform aspect practices

The purpose of this generic practice is to ensure that aspect practices are achieved. Aspect practices describe fundamental software development activities without emphasis on agile practices. Agile elaboration for each aspect practice is also given to provide a roadmap, however, it is not expected to fully achieve the agile elaborations.

Generic Resources for "Performing Aspect Practices" Attribute

— Resources that are used to perform aspect practices.

Generic Work Products for "Performing Aspect Practices" Attribute

— Ref: Work products at aspect dimension

A.1.3. Agility Level 2: Lean

Aspects at Agility level 2 are characterized with two attributes: "Iterative" and "Simple". Work products are developed *iteratively* so that frequent feedback are provided to and obtained from customer to improve the product capabilities, to employ variability, to identify bottlenecks and problematic areas and to verify assumptions related to product.

"Simple" aspects' major focus is to deliver business value as quickly as possible by eliminating non-value added activities. By being *simple,* aspects will have the agility to embrace change and adapt changing conditions, since the impact of the change on the currently developed work products is minimum.

A "Lean" aspect simply means that with its all practices, it is optimized to deliver software fast.

If we look at from a broader point of view; teams need to internalize a communication oriented culture to obtain the outcomes of iterative development and simplicity.

By performing generic practices of iterative and simple attributes the following outcomes are achieved:

- Having a quick learning ability as a result of frequent feedback
- Preventing errors earlier in the life cycle
- Increased visibility of teams
- Response capability to business needs
- Embracing change
- Managing changing priorities
- Adaptation to changing business conditions
- Increased communication Agility
- Eliminating non-value added activities from the system

A.1.3.1. Aspect Attribute 2.1 "Iterative"

"Iterative" attribute is a measure of the extent to which the work products are delivered in an iterative and incremental way to achieve the following outcomes:

As a result of full achievement of this attribute:

a) Customer is satisfied with early and continuous valuable software [31].
b) Working software is delivered frequently with short time scales [31].
c) Frequent feedback from customer is obtained.
d) Bottlenecks, emergencies are discovered as quickly as possible.
e) Assumptions made throughout the whole process are verified.
f) Wrong implementations are realized early, bad consequences of the decisions are minimized.
g) A trusted environment is created so that business people and development team are able to work together throughout the whole project [31]

Generic Practices for "Iterative" Attribute

GP 2.1.1 Develop work products in an iterative and incremental way

— Whole software development cycle is performed in multiple iterations
— The numbers of the iterations are determined (updated) during development of the project based on the feedback obtained.
— High quality working software is demonstrable with every iteration. If it is not, it means that everything done during the development process is not adding value to software [11]

GP 2.1.2 Communicate effectively

— Communication interfaces are established between internal, external stakeholders and the team to obtain fast and continuous feedback and to improve shared understanding.
— Customer is involved in the software development process
— Various communication tools such as information radiators, white boards or walls, projectors, team rooms etc. are utilized during the iterations
— Daily meetings, planning meetings, Kanban boards, backlogs, and requirements are used as communication channels between the team itself and the team and internal and external stakeholders.
— A trusted environment relying on transparency and respect is created.
— Positive and negative feelings can be talked between team members.

Generic Resources for "Iterative" Attribute

— Internal Stakeholders (business owners, managers, subject matter experts...)
— External Stakeholders (customers, user...)
— Product owner, process manager (scrum master)
— Team Members
— Communication environment (intranet links, rooms to communicate)
— Tools for technical support (such as requirements management tool, task management tool, configuration management tool, backlog management tool).

Generic Work Products for "Iterative" Attribute

— Working Software
— Iteration Plans
— Communication Notes
— Test Results
— Burn Down Charts
— KanBan Boards
— Customer Responses

A.1.3.2. Aspect Attribute 2.2 "Simple"

This attribute is a measure of the extent to which the aspect practices are arranged and performed by focusing on delivering business value. The purposes of "simple" attribute are to support aspects to eliminate any kind of activity that does not add value and cause waste in software development process, to achieve the balance between the just-in-time works and up-front works and to manage the incoming and outgoing workflows.

By eliminating waste, balancing work products and workflow and identifying problems as early as possible, team will adapt to changes and deliver business value as quickly as possible.

As a result of full achievement of this attribute;

a) Waste (waiting, partially done work, over-production, over-processing…) is eliminated
b) Up-front predictive and just-in-time adaptive work are balanced
c) Cost of the change is kept at an acceptable level even it is identified late in the process
d) Changes are embraced in every stage of software development life cycle

Generic Practices for "Simple" Attribute

GP 2.2.1 Balance the predictive work and adaptive work

- The balance between the up-front work and just-in-time work is achieved.
- The flow of the work is balanced: Development speed of the work products are arranged so that speed of the incoming items are equal to or larger than the outgoing items
- End uncertainty (what to build), means uncertainty (how to build) and customer uncertainty (for whom to build) are reduced together with a holistic approach.
- Low-cost explorations are used to gather required information (obtain feedback) and to reduce the need for assumptions
- Decisions are made as late as possible.

GP 2.2.2 Employ minimally sufficient ceremony

- Heavyweight, not value added practices are eliminated; process heavy document-oriented work is minimized (day long reviews that lock the requirements activities, non-value added records kept in tools, long process flows implemented on tools)
- Criteria to write documents are identified

Generic Resources for "Simple" Attribute

- Human resources: Backlog manager, process manager, customer
- Tools for technical support: requirements management tool, task management tool, configuration management tool, backlog management tool.

Generic Work Products for "Simple" Attribute

- Product Backlog/Backlog
- Sprint Backlog
- Incremental Design
- Daily Builds

A.1.4. Agility Level 3: Effective

Agility level three is called “Effective” where each aspect is performed to achieve delivering value with high productivity and low defects by employing agile engineering practices and using agile tools for support in a continuously improving environment. The *Effective Level* are characterized with two attributes: “Technical Excellence” and “Learning”.

Agile engineering practices such as test-driven development, continuous integration, and pair programming and integration of agile tools bring technical excellence to aspects. When technical excellence and other attributes from second level are brought together, teams gain the Agility to manage technical debt, improve team productivity and decrease defects.

By performing generic practices of the attributes, the following outcomes are achieved:

- Agility to manage technical debt
- Expertise at agile engineering practices
- Delivering value with high morale and excellence (in terms of high productivity and low defect rates) as a result of technical expertise
- Continuous improvement

A.1.4.1. Aspect Attribute 3.1 Technically Excellent

This attribute is a measure of the extent to which the agile engineering methods and tools are integrated into aspects to improve productivity and lower defects.

To reach technical excellence, people invest on learning and practicing new engineering methods/practices and implementing these practices on real world examples [102]. This requires a significant amount of time and determination before obtaining the benefits.

Technical excellence cause changes in culture of agile teams. This jump means moving from giving quick responses to dynamically changing conditions to leveraging the balance between response time and quality [11]. Perception of the whole team and other involving stakeholder move to balance and perfection stage.

As a result of full achievement of this attribute:

a) Aspects, practices, team, quality, productivity are kept improving
b) Response agility and productivity of the team is increased
c) Defects are reduced
d) Everyone is able to access to the same information with the help of tools
e) Progress is made visible to whole team and other stakeholders

Generic Practices for “Technically Excellent” Attribute

GP 3.1.1 Incorporate agile engineering methods/practices to the aspects

— Engineering methods/practices such as test-driven development, continuous integration, code review, pair programming and others are performed

GP 3.1.2 Integrate tools to aspects to improve the productivity

— Requirement management tools, agile tracking tools, configuration management tools are integrated

Generic Resources for "Technically Excellent" Attribute

— Trainers for technical skills
— Tools for management, integration, analysis, etc…

Generic Work Products for "Technically Excellent" Attribute

— Outputs of engineering practices

A.1.4.2. Aspect Attribute 3.2 "Learning"

"Learning" attribute is a measure of the extent to which from a broader point of view aspects serve for the purpose of organizational learning and improvement

As a result of full achievement of this attribute:

a) Individuals learn from each other, share knowledge and improve together
b) Individuals on the team share the responsibility of the conduct of the aspect practices and improvement of aspects
c) Better aspects, better team structures, better environment are achieved.
d) Productivity, agility and quality attributes are improved.

Generic Practices for "Learning" Attribute

GP 3.2.1 Support collaborative work and shared responsibility

— Collaborative work
— Self-organizing teams
— Shared responsibility
— Specializing in more than one area
— Decision making by using collective experiences

GP 3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people

— Being inspiring leaders for people to fulfill their potential instead of commanding, controlling and task assigning managers
— Learning from mistakes instead of blaming and assigning the responsibility to the person making the mistake

GP 3.2.3 Encourage people in the organization to participate in learning, teaching and improvement

— Learning and continuous improvement at team level and organizational level are achieved
— People learn from each other
— Knowledge is shared throughout the organization

GP 3.2.4 Collect measures to support learning and improvement

- Measures, objectives of measures, boundary limits and data collection procedures are specified.
- Measures are collected and analyzed regularly

Generic Resources for "Learning" Attribute

- All team members
- Internal Stakeholders (business owners, managers, subject matter experts...)
- External Stakeholders (customers, user...)
- Learning platforms
- Shared resource platforms

Generic Work Products for "Learning" Attribute

- Feedback obtained from customer
- Emerging ideas (improvement suggestion) list
- Training materials
- Action item list

A.2. Aspect Dimension

A.2.1. Exploration Aspect

Purpose1: The purpose of the exploration aspect is to understand the customer/user needs and transform these needs into artifacts that initiate communication for elaboration on them during the construction and manage the changes in these artifacts.

Outcomes:

1. Customer and user needs that represent the characteristics of a software product are captured.
2. The needs are elaborated and transformed into requirement artifacts at various levels of abstraction (Agile Elaboration (AE): themes, epics, features, user stories, technical stories, use cases etc...).
3. Conflicts in requirements artifacts are detected and resolved.
4. Artifacts are prioritized for construction and changes to these artifacts are managed.
5. Artifacts are made visible to everyone to create collaboration and transparency.

Exploration Aspect Practices:

E.AP1: Capture the customer and user needs: Perform requirement envisioning activities with customer/user to obtain tacit knowledge and to capture functional and non-functional requirements as high level work items (AE: user needs can be obtained as themes, epics or features. Involving users, customers to the team in determining what to build and reviewing what is being built is a better option than scheduled meetings [11]). [Outcome 1-2]

E.AP2: Elaborate requirements artifacts: Elaborate high level requirements artifacts into required level of detail for construction. (AE: Elaborate themes, epics, features into stories for further detail by communicating with the customer/user/team members.) [Outcome 2]

E.AP3: Detect and resolve conflicts of requirements artifacts: Detect and resolve conflicts related to requirements artifacts (AE: prefer high bandwidth communication techniques) (Outcome 3)

E.AP4: Specify dependencies among requirements artifacts: Detect and specify dependencies among stories and other artifacts to prevent a failure caused by a missed dependent requirement

E.AP5: Manage the requirement artifacts: Manage the change on requirements artifacts. Identify the impact of the change on artifacts and project timeline. (AE: agile conformant work item management strategies such as a product backlog or a work item stack can be utilized. Prioritize work items and communicate on the changed items with the team, update the backlog and re-prioritize the backlog items for change management). (Outcome 4)

E.AP6- Make the artifacts visible to everyone: Make the requirement artifacts (AE: backlog items) visible to everyone in order to create collaboration and transparency. (Outcome 5)

Example Work Products

- Requirement specifications, user requirements, software requirements
- Work items on requirement management tools
- Change records
- Traceability records
- (AE: Backlog filled in with agile requirements at different level of detail: themes, epics, features, user stories, technical stories, use cases)

AE: Fallacies of agile practices:

F1: Lightweight story descriptions are only acceptable when they are supported with communication in which developers, testers and customer (or customer representatives) are involved.

F2: Backlog may contain items that are already elaborated or need more elaboration before implementation begins. Define the items goal-oriented and focus on "what" rather than "how". For team backlog keep the items big enough for the team to complete in a single iteration. Assure that the items are independent, negotiable, valuable; estimable, small and testable.

F3: If stories are utilized as a way of transition of the customer needs to development team; then stories are better to conform INVEST criteria (Independent, Valuable, Estimable, Small and Testable) [11]**.**

A.2.2. Construction Aspect

Construction aspect includes architecture, design, coding and unit testing activities.

Purpose: The purpose of the construction aspect is to develop a high-quality software solution that is ready to be built.

Outcomes:

1. Work items are elaborated to initiate building a solution
2. Architecture of the system is developed/maintained (AE: architecture is continuously evaluated)
3. Solution is designed to meet the customer requirements
4. Selected solution is developed
5. Correctness of the software units are ensured
6. Test environment is established or maintained based on the test system requirements.

Construction Aspect Practices:

CN.AP1: Elaborate the work items: Before building a solution, capture detailed information about the work items. (AE: In agile projects, these work items are selected from the backlog. They can be either a user story, defect, technical spike or else. Detailed information may be captured in either writing executable specifications (acceptance tests-"tests as requirements"), either performing high level specifications, or either collaborative just–in-time modeling or else). (Outcome 1)

CN.AP2: Explore the design: Design the solution before building it, to avoid technical debt. Explore the solution for both of the functional and non-functional requirements/stories. (AE: Use alternative design approaches such as model storming (just in time brainstorming to explore alternative designs over models), architecture prototyping, user interface design, model-driven development (allows the UML models and the code be synchronized) and test-driven development (allows design refactoring)[1] Perform technical search, dependency identification, and risk evaluation activities in addition to design). (Outcome 2-3)

CN.AP3: Develop the solution: Develop the software that conforms to the requirements. (Outcome 4)

CN.AP4: Ensure the correctness of software at developer level: Test the correctness of the software units. (AE: Software units are tested in an automated way before the integration. Automated unit test suites are constructed. Automated tests are triggered whenever a change is introduced to the software) [103]. Quality of the code is enhanced by adding static code analysis checks, reviews and inspections. Code is refactored regularly to avoid technical dept.) (Outcome 5-6)

Example Work Products

- Source Code
- Component and component integration test results
- Unit Test Scripts
- Domain Mapping Matrix
- Dependency Structure Matrix
- Architecture Scheme of the System

— Architecture Tradeoff Analysis Results
— Code Review Results

Fallacies for agile practices:

Proofing the architecture with working code is a key element in building a qualified solution. Paper-based architecture design is not a substitute for the working code [1]

A.2.3. Transition Aspect

Purpose: The purpose of the transition aspect is to establish and maintain reliable and repeatable build, integration and deployment practices to keep the application in a working state during the development, to obtain feedback about the problems in the process, to make the whole process visible to everyone, and to shorten the response time to changes.

Outcomes:

1. Testing, deployment and release activities are integrated into the development process [103]
2. Test and production environments, source code, operating systems, 3rd party elements, libraries and any kind of patches are kept under configuration control.
3. Changes are integrated to code (AE: Continuously and rapid feedback are obtained about the changes committed and the problems occurred.)
4. Software is deployed to different environments (AE: Deployment is done continuously.)
5. Developed solution is verified. (AE: Evaluation is performed through automated tests and automated code analysis techniques. Defects are identified and fixed where they are introduced and bug accumulation is prevented.)
6. AE: Transparency of the process among all stakeholders is ensured.
7. Documents to maintain the software are produced.

Aspect Practices:

T.AP1: Create and Manage the Workspace: Create an environment that the source code, database scripts, test data, build scripts, 3rd party libraries and deployment scripts are taken under configuration control and have the latest version. Record every change that has been made in the source code, data and testing and production environments. Prevent manual changes to these items. [Outcome 1-2]

T.AP2: Integrate the Code: Integrate the code and "build and test" your application. (AE: Integration is performed frequently. The application is built and tested automatically with every check-in to obtain rapid feedback about the changes that are committed [103]. The techniques that enable the build of the whole system can be done with a single command within minutes are adopted.) (Outcome 3)

T.AP3: Deploy the solution: Ensure that the builds are deployed to various environments and the target environment is running correctly after deployment. (AE: Deployment is performed continuously and automatically) (Outcome 4)

T.AP4: Test the integrated solution: Test the integrated software for both functional and nonfunctional requirements. (AE: Automated test suites at regression and acceptance test levels are created. Software is tested through automated tests. Manual tests are also performed for exploration and usability testing purposes. High test coverage is ensured during the verification and validation.) (Outcome 5)

T.AP5: Make the progress visible: (AE: Make the transition process visible to everyone who are involved in the process to improve transparency and collaboration) (Outcome 6)

T.AP6: Create the supporting documentation: Create and deliver the support documentation to the stakeholders. Decide the amount of the documentation for negotiation with the team and the external stakeholders. Produce the documents required to maintain the software. (Outcome 7)

Example Work Products:

- Integrated and working code
  Detected Bugs
  Build Scripts
  Test Scripts
- Test Data
- Bug Records
- Test Logs
- Support documentation

Fallacies for agile practices:

F1: Using Branches: Using branches in version control does not recommended in continuous integration. Because, while working on a branch, the code is not being integrated with other developers for a long time that will definitely cause integration problems described in F3 [103].
F2: Automated nightly builds or building only when it is demanded are not considered continuous integration. Automated nightly build can be a good start on the way of continuous integration process.
F3: Keeping the release cycle long: When the deployment is done after the whole development is finished, development team needs more time to find and fix bugs and incorrect assumptions on features remains undetected for a long time
F4: Not having an automated test suite: Passing a build without the support of an automated test suite means that the software could be compiled and assembled [103]. However, it does not give you the confidence of software working properly.

A.2.4. Management Aspect

Purpose: The purpose of management aspect is to identify, establish and track activities and resources necessary to develop a product. (AE: From agile perspective, the purpose of the management aspect is to perform planning and tracking activities continuously, and estimating collaboratively to achieve efficiency and perform these practices as value adding activities to the project life cycle.)

Outcomes

1. Feasibility studies are performed, product vision and scope are established.
2. Teams are formed, environment for development is established.
3. Activities are planned throughout the project's lifecycle based on the information at hand (AE: Continuous planning is performed)
4. Work items are estimated. (AE: Estimation is done in a collaborative environment)
5. (AE: Estimation is validated through whole team)
6. Velocity of the team are calculated dynamically in an adaptive way, status information is updated regularly.
7. Risks are monitored and preventive and corrective actions are taken.

Aspect Practices

M.AP1: Initiate the project: Perform a feasibility study and present the product vision and scope to bring the product dimension into alignment. Decide sufficient levels of documentation and tailor the practices based on the characteristics of the project. (**AE**: Use simple and low ceremony practices for project initiation activities such as the "vision box" for the vision and "project data sheet" for the project scope [25]. Improve the product vision based on new information and obtain agreement of the team and other stakeholders about the vision.) (Outcome 1)

M. AP2: Form the team: Form the team with right people who are from different backgrounds and are capable of developing the software product collectively with sufficient knowledge and experience. (Outcome 2)

M.AP3: Align with agile values and principles: (**AE**: Align internal and external stakeholders with agile values and principles.) (Outcome 2)

M. AP4: Establish the physical workspace: (**AE**: Construct an open space where each team member can communicate easily with each other, construct quiet and private places, support usage of information radiators, allow outlets for fun to gain the energy back.) (Outcome 2)

M.AP5: Plan the progress: Identify activities, tasks, resources and required experience, knowledge and skills of the team members. Define dependencies among activities and tasks. (**AE**: Establish a high level feature based plan at the initiation phase of the project. Define approximate numbers of iterations to complete the project. Elaborate the plan with each iteration considering the prioritized list of user stories/use cases. Coordinate daily activities with the team. Ensure that each of these activities provide value. Let the team to participate in planning so that each team member reflects his/her expertise and experience) (Outcome 3)

M.AP6: Estimate the work items: Estimate the size, effort and schedule for the given requirements of the project. (AE: Estimate the size of user stories, themes, epics, or any piece of work in hand. Perform a whole team estimation to improve the accuracy of estimates. Add people to estimation process who are competent in solving the problem

and encourage discussions in a collaborative environment. Re-estimate the feature when its relative size changes.) (Outcomes 4-5)

M.AP7: Monitor the progress: Monitor the progress of the projects through project parameters (effort, schedule, cost, etc.) (**AE:** Track the progress of the activities through daily and weekly team integration activities. Define the team's rate of progress (velocity) for iterations from past experiences in terms of size. Update project dashboard (status of the work items, estimation and velocity updates) regularly with the new information arrives.) (Outcome 6)

M.AP8: Manage and mitigate the risks: Identify project risks, analyze, prioritize and track the status of them regularly. Make contingency and mitigation plans based on the priorities of the risks. (Outcome 7)

Fallacies for agile practices:

F1: Larger and vague features, epics, user stories leads to uncertain estimates, however, one shouldn't wait for the estimation until detailed elaborations are performed. **F2:** Size estimation shouldn't be confused with the estimation of the amount of time it takes to implement a feature**.**

F3: Don't re-estimate when the team thinks that they should have completed more points to increase velocity, when less "done" is completed.

F4: Create plans that focus on the delivery of the features that value to customer rather than completion of the activities. Avoid from activity based plans.

F5: Extension of the delivery dates indicates that there might be a problem in estimation or in delivering commitments

F5: Daily stand-up meetings which are limited to a short time are not optional since they ensure transparency and accountability among team members.

F6: Team member’s sign-up for additional tasks by themselves when finishing their tasks and not waiting for the scheduled time is up

APPENDIX B

Agility Assessment Report of Organization N, Project #1

1. EXPLORATION ASPECT

EXPLORATION ASPECT			
Aspect Practices	**Current Application**	**Improvement Suggestions**	**Rating**
E.AP1: Capture the customer and user needs: Perform requirement envisioning activities with customer/user to obtain tacit knowledge and to capture functional and non-functional requirements as high level work items. (AE: user needs can be obtained as themes, epics or features. Involving users, customers to the team in determining what to build and reviewing what is being built is a better option than scheduled meetings [11])	Advertisement team, content provider team business intelligence team and end users are the sources of requirements. Requirements are gathered in regular meetings where the feedback obtained from real system are discussed and new trends are evaluated. The requirements are recorded to the meeting records.	Obtained business needs and requests should be recorded to a list or a system where a unique number is assigned each of the items	2
E.AP2: Elaborate requirements artifacts: Elaborate high level requirements artifacts into required level of detail for construction. (AE: Elaborate themes, epics, features into stories for further detail by communicating with the customer/user/team members.)	Analysts and content providers work closely in a collaborative environment. Requirements are elaborated, workflow diagrams are developed when needed.	Elaborated requirements should be added to the "requirements list" keeping all relations.	2

E.AP3: Detect and resolve conflicts of requirements artifacts: Detect and resolve conflicts related to requirements artifacts. (AE: prefer high bandwidth communication techniques)	Conflicts are resolved through communication channels among team members.		3
E.AP4: Specify dependencies among requirements artifacts: Detect and specify dependencies among stories and other artifacts to prevent a failure caused by a missed dependent requirement	Dependency of a new or changed requirement to other requirements are defined based on the personal experiences of team members. Dependency identification process are person-dependent in organization N.	Evaluation and identification of dependencies among requirement items are essential to manage changes properly and embrace change. Dependencies are needed to be identified in a relational matrix or on the system where requirements are defined.	1
E.AP5: Manage the requirement artifacts: Manage the change on requirements artifacts. Identify the impact of the change on artifacts and project timeline. (AE: agile conformant work item management strategies such as a product backlog or a work item stack can be utilized. Prioritize work items and communicate on the changed items with the team, update the backlog and re-prioritize the backlog items for change management).	Grooming activity are performed regularly at the end of each week for high level requirements. All parties are informed if there is a change on the requirements and requests.		3
E.AP6- Make the artifacts visible to everyone: Make the requirement artifacts (AE: backlog items) visible to everyone in order to create collaboration and transparency.	Some of the requirements are made visible by application of Kanban Board.	In addition to Kanban Board, the whole prioritized requirement list should be made visible to Project team.	2

EXPLORATION Aspect – LEAN LEVEL Assessment			
Iterative Attribute			
GP 2.1.1 Develop work products in an iterative and incremental way	Identification of user needs and elaboration are performed in an incremental and iterative way. The cycle time for each iteration are 7 to 10 days. Iterations are regular and consistent.		3
GP 2.1.2 Communicate effectively	Communication interfaces are established between internal, external stakeholders. A real Kanban Board is utilized for all team members come together and communicate on daily activities.		3
Simple Attribute			
GP 2.2.1 Balance the predictive work and adaptive work	High-level requirements are classified to phases and each requirement are elaborated and detailed with every iteration. Flow of the requirements are balanced.	A consistency in definition of the requirements are needed to be achieved. User stories or use cases can be preferred.	2
GP 2.2.2 Employ minimally sufficient ceremony	Informal procedures are applied for the approval of requirements.	Criteria to write requirement documents are needed to be identified. For the identification of non-value added activities, regular retrospective studies are needed to be performed.	1

EXPLORATION A. EFFECTIVE LEVEL Assessment			
Technical Excellence Attribute			
GP 3.1.1 Incorporate agile engineering methods/practices to the aspect practices		A backlog type structure is needed to record all business needs and other issues for development. Requirements should be included to this backlog. A unique number should be assigned to each request or business need. The backlog should describe "what" the system does rather than "how" Backlog are needed to be kept in a prioritized order where every member of team could see. Requirements should satisfy INVEST criteria. (Independent, Value-Added, Small, Estimable, Testable)	1
GP 3.1.2 Integrate tools to aspects to improve the productivity		Tools such as Jira, VersionOne, MSF for Agile are needed to be included in the conduct of the aspect.	1
Learning Attribute			
GP 3.2.1 Support collaborative work and shared responsibility	Requirements are analyzed and elaborated in a collaborative environment, team shares the whole responsibility.		3

GP 3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people	When a problem or error occurred, team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error.		3
GP 3.2.3 Encourage people in the organization to participate in learning, teaching and improvement	People learn and improve by their own efforts. They are not trained about agile approaches or practices. Lessons learned are not evaluated or kept.	It is very beneficial to have an organizational training strategy and plan. Internal and external trainings are needed to be planned to facilitate agile adoption process. A system to record lessons learned on requirement activities are needed to be established. Lessons learned are needed to be shared to whole organization.	1
GP 3.2.4 Collect measures to support learning and improvement	A comprehensive infrastructure had been established to track measures real time from the running system such as number of videos viewed, video upload time etc. However, exploration aspect's activities are not followed through measures just direct observation.	A measurement infrastructure are needed to be established to monitor the progress and identify improvement areas related to the exploration aspect.	1

2. CONSTRUCTION ASPECT

CONSTRUCTION ASPECT			
Aspect Practices	**Current Application**	**Improvement Suggestions**	**Rating**
CN.AP1: Elaborate the work items: *Before building a solution, capture detailed information about the work items* (*AE: In agile projects, these work items are selected from the backlog. They can be either a user story, defect, technical spike or else. Detailed information may be captured in either writing executable specifications (acceptance tests-"tests as requirements"), either performing high level specifications, or either collaborative just–in-time modeling or else).*	Details about high level requirements are revealed by developers, mock-up screens are developed. Business people are also involved in these studies.	When the details are obtained about requirements, backlog should be updated.	2
CN.AP2: Explore the design: *Design the solution before building it, to avoid technical debt. Explore the solution for both of the functional and non-functional requirements/stories. . (AE: Use alternative design approaches such as model storming (just in time brainstorming to explore alternative designs over models), architecture prototyping, user interface design, model-driven development (allows the UML models and the code be synchronized) and test-driven development (allows design refactoring) [1] Perform technical search, dependency identification, and risk evaluation activities in addition to design).*	Alternative designs are evaluated, functional dependencies are considered.	Alternative designs should be evaluated not just for the graphical user interfaces, but for the architecture. Dependencies between design elements, impact of new requirements on current design elements are needed to be identified. (Dependency Structure Matrixes) Team should ensure that to be developed solution (design)	2

		should meet functional and non-functional requirements.	
CN.AP3: Develop the solution: *Develop the software that conforms to the requirements.*	PHP language is utilized for coding. Comments are added to code to specify changes, but not regularly.	Code should be commented regularly for maintenance activities.	2
CN.AP4: Ensure the correctness of software at developer level: *Test the correctness of the software units. Establish the test environment and maintain it. (AE: Software units are tested in an automated way before the integration. Automated unit test suites are constructed. Automated tests are triggered whenever a change is introduced to the software) [103]. Quality of the code is enhanced by adding static code analysis checks, reviews and inspections).*	Developers verifies the code over GUI. Code is reviewed occasionally.	Coding standards are needed to be defined and applied. The best and easiest way to ensure application of coding standards is to add static code analysis checks to process. The efficiency of the developer tests are needed to be evaluated. Success criteria to be achieved before moving the code testing phase should be defined.	2
CONSTRUCTION Aspect – LEAN LEVEL Assessment (2nd Level)			
Iterative Attribute			
GP 2.1.1 Develop work products in an iterative and incremental way	Coding activities are performed iteratively and incrementally.		3
GP 2.1.2 Communicate effectively	Communication interfaces are established between team members and other stakeholders. Team members come together discuss daily activities through daily meetings.		3
Simple Attribute			

GP 2.2.1 Balance the predictive work and adaptive work		Code should be reviewed before testing. In agile development it is suggested that every team member could be involved in testing activities by coding unit tests.	2
GP 2.2.2 Employ minimally sufficient ceremony		An efficient approach to evaluate dependencies among design elements should be defined.	2
CONSTRUCTION A. EFFECTIVE LEVEL Assessment (3rd Level)			
Technical Excellence Attribute			
GP 3.1.1 Incorporate agile engineering practices to the aspect practices		Unit test should be written in terms of developer testing to obtain rapid feedback. It is suggested to apply test-driven development approach to construct a more reliable system. Shortcut solution to solve emergent problems create burden on code and cause technical dept. To avoid it code are needed to be regularly refactored.	1
GP 3.1.2 Integrate tools to aspects to improve the productivity	GIT tool and check-in, check-out mechanisms are being used for version control.	Unified Modelling Language could be preferred for design development.	2

135

Learning Attribute			
GP 3.2.1 Support collaborative work and shared responsibility	Collaboration among team members are seen in development activities as well. Developers select their own tasks to work. Development team leader leads the team in technical issues.		3
GP 3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people	When a problem or error occurred, team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error.		3
GP 3.2.3 Encourage people in the organization to participate in learning, teaching and improvement	People learn and improve by their own efforts. They are not trained about agile approaches or practices. Lessons learned are not evaluated or kept.	A knowledge management system should be established to share learned knowledge and organizational memory covering coding activities.	1
GP 3.2.4 Collect measures to support learning and improvement	Construction aspect's activities are not followed through measures just direct observation.	Agile metrics are needed to be defined and collected for construction activities.	1

3. TRANSITION ASPECT

TRANSITION ASPECT			
Aspect Practices	**Current Application**	**Improvement Suggestions**	**Rating**
T.AP1: Create and Manage the Workspace: *Create an environment that the source code, database scripts, test data, build scripts, 3rd party libraries and deployment scripts are taken under configuration control and have the latest version. Record every change that has been made in the source code, data and testing and production environments. Prevent manual changes to these items*	Code is under configuration control. There are two environment: "development" and "production". However, for coding and testing activities same environment is being used. Check-in, check-out mechanisms are being used for labeling and versioning.	Uncontrolled change is performed on the code. A system should be established in order to link requirements or tasks to changesets or unique numbers of the requirements or tasks can be in the code as comments. Version control systems enables the attachment of code to the requirement items.	2
T.AP2: Integrate the Code: *Integrate the code and "build and test" your application. AE: Integration is performed frequently. The application is built and tested automatically with every check-in to obtain rapid feedback about the changes that are committed [103]. The techniques that enable the build of the whole system can be done with a single command within minutes are adopted*	Code sets are being integrated in the "development" environment after they are developed in developers' local computers.		3
T.AP3: Deploy the solution: *Ensure that the builds are deployed to various environments and the target environment is running correctly after deployment. (AE: Deployment is performed continuously and automatically)*	Code is automatically deployed to the "production" environment. This is a facility of PHP language.		3

T.AP4: Test the integrated solution: *Test the integrated software for both functional and nonfunctional requirements. (AE: Automated test suites at regression and acceptance test levels are created. Software is tested through automated tests. Manual tests are also performed for exploration and usability testing purposes. High test coverage is ensured during the verification and validation.)*	Test engineer has a right to access to the "development" environment via a Proxy code and tests the application via graphical user interfaces.	Test preparation activities should start at the same time with development activities. Bugs should be recorded and retested or unit tests should be written immediately where a bug is found to eliminate manual testing and increase test coverage.	1
T.AP5: Make the progress visible: *(AE: Make the transition process visible to everyone who are involved in the process to improve transparency and collaboration)*		All transition activities are needed to make visible to whole team.	2
T.AP6: Create the supporting documentation: *Create and deliver the support documentation to the stakeholders. Decide the amount of the documentation for negotiation with the team and the external stakeholders. Produce the documents required to maintain the software.*		Supporting documentation should be developed for the maintenance of the system or essential information should recorded on the software tool where information is available when needed.	2
TRANSITION Aspect – LEAN LEVEL Assessment			
Iterative Attribute			
GP 2.1.1 Develop work products in an iterative and incremental way	Transition aspect activities are performed iteratively and incrementally.		3
GP 2.1.2 Communicate effectively	Team members share the same room and effectively communicate.	Information radiator that is usually utilized to specify the	2

		person who commits to the mainline and the status of integration can be used.	
Simple Attribute			
GP 2.2.1 Balance the predictive work and adaptive work		To balance the predictive work and adaptive work test preparation activities should be performed. Test cases are needed to be specified. (It can be started with regression test cases) Another suggestion to achieve the balance is to start coding activities be developing the tests. Status of the integration can be followed through integration tools	1
GP 2.2.2 Employ minimally sufficient ceremony	Informal procedures are applied for the approval of transition decisions	For the identification of non-value added activities, regular retrospective studies are needed to be performed.	2
TRANSITION A. EFFECTIVE LEVEL Assessment			
Technical Excellence Attribute			
GP 3.1.1 Incorporate agile engineering methods/practices to the aspect practices		If development and testing activities are performed in the	1

		same environment TDD approach is suggested. Another option is to separate development and test environments. Continuous integration (code frequently committed to the mainline. Whenever a code checks-out all code are compiled and all automated tests are run) are needed to obtain rapid feedback and keep the mainline always in a working state. Automated test suite are needed to be developed.	
GP 3.1.2 Integrate tools to aspects to improve the productivity	GIT tool is being used for integration and transition activities.		2
Learning Attribute			
GP 3.2.1 Support collaborative work and shared responsibility	Collaboration among team members are seen in integration and deployment activities.		3
GP 3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people	When a problem or error occurred, team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error.		3

GP 3.2.3 Encourage people in the organization to participate in learning, teaching and improvement	People learn and improve by their own efforts. They are not trained about agile approaches or practices. Lessons learned are not evaluated or kept.	A knowledge management system should be established to share learned knowledge and organizational memory covering transition activities.	1
GP 3.2.4 Collect measures to support learning and improvement	Transition aspect's activities are not followed through measures just direct observation.	Agile metrics are needed to be defined and collected for transition activities.	1

4. MANAGEMENT ASPECT

MANAGEMENT ASPECT			
Aspect Practice	**Current Application**	**Improvement Suggestions**	**Rating**
M.AP1: Initiate the Project: *Perform a feasibility study and present the product vision and scope to bring the product dimension into alignment. Decide sufficient levels of documentation and tailor the practices based on the characteristics of the project. (AE: Use simple and low ceremony practices for project initiation activities such as the "vision box" for the vision and "project data sheet" for the project scope [25]. Improve the product vision*	It is said that the project vision is known by everyone in the team. The scope changes based on technological improvements and analysis.	Even if the vision is known by every team member, it is beneficial to define Project vision and update it based on technical improvements.	2

based on new information and obtain agreement of the team and other stakeholders about the vision.)		Documentation that are needed to be developed should be agreed at the beginning of the Project. Vision box, project data sheet practices can be used for vision and scope definitions.	
M.AP.2: Form the team: *Form the team with right people who are from different backgrounds and are capable of developing the software product collectively with sufficient knowledge and experience.*	Knowledge, capability and experience of the employee are the critical factors for being a team member.		3
M.AP.3: Align and adopt the environment: *(AE: Align internal and external stakeholders with agile values and principles.)*	Internal and external stakeholders are familiar to the idea of agile software development. However, there is no consensus in the organization yet.	Both internal and external stakeholders are needed to be informed about the agile values, a consensus should be established.	1
M.AP.4: Establish the physical workspace: *(AE: Construct an open space where each team member can communicate easily with each other, construct quiet and private places, support usage of information radiators, allow outlets for fun to gain the energy back.)*	Physical workspace conditions overlap with agile suggestions.	Quite workspaces can be arranged for the team members who needs privacy.	3
M.AP.5: Plan the progress: *(AE: Establish a high level feature based plan at the initiation phase of the project. Define approximate numbers of iterations to complete the project. Elaborate the plan with each iteration considering the prioritized list of user stories/use cases.*	High-level plans are elaborated with every iteration. Daily activities are coordinated through Daily stand-up meetings.		3

Coordinate daily activities with the team. Ensure that each of these activities provide value. Let the team to participate in planning so that each team member reflects his/her expertise and experience)			
M.AP.6: Estimate the work items: Estimate the size, effort and schedule for the given requirements of the project. (AE: Estimate the size of user stories, themes, epics, or any piece of work in hand. Perform a whole team estimation to improve the accuracy of estimates. Add people to estimation process who are competent in solving the problem and encourage discussions in a collaborative environment. Re-estimate the feature when its relative size changes.)	Effort estimation is performed for requirements. Each member estimate his/her own task's effort based on past experiences.	Size needs to be estimated besides effort estimation. An estimation approach is needed to be established.	2
M.AP.7: Monitor the progress: *Monitor the progress of the projects through project parameters (effort, schedule, cost, etc.) (AE: Track the progress of the activities through daily and weekly team integration activities. Define the team's rate of progress (velocity) for iterations from past experiences in terms of size. Update project dashboard (status of the work items, estimation and velocity updates) regularly with the new information arrives.)*	Project managers track the project over the numbers of closed tasks.	It is not just to follow up the Project over completed tasks. Other parameters such as team velocity, the size of completed and remaining tasks are needed to be tracked.	2
M.AP.8: Manage (mitigate) the risks: *Identify project risks, prioritize and track the status of them regularly. Take preventive and corrective actions*	Project risks are not tracked, corrective actions are taken after a risk occurs.	Risk monitoring approaches are needed to be evaluated.	1
MANAGEMENT Aspect – LEAN LEVEL Assessment			
Iterative Attribute			

GP 2.1.1 Develop work products in an iterative and incremental way	Management aspect activities are performed iteratively and incrementally.		3
GP 2.1.2 Communicate effectively	Team members share the same room and effectively communicate.		3
Simple Attribute			
GP 2.2.1 Balance the predictive work and adaptive work	Decision are taken collaboratively, team shares the responsibility. The balance is achieved for the management activities		2
GP 2.2.2 Employ minimally sufficient ceremony	Informal procedures are applied for the approval of transition decisions	For the identification of non-value added activities, regular retrospective studies are needed to be performed.	2
MANAGEMENT A. EFFECTIVE LEVEL Assessment			
Technical Excellence Attribute			
GP 3.1.1 Incorporate agile engineering methods/practices to the aspect practices	Effort is estimated.	Agile approaches are needed to be evaluated for size and effort estimation.	1
GP 3.1.2 Integrate tools to aspects to improve the productivity	Asana tool and Kanban board is utilized for management activities.	The information kept in real Kanban board can be transferred to management tools that provide effective management and tracking over metrics.	2
Learning Attribute			
GP 3.2.1 Support collaborative work and shared responsibility	Collaboration among team members are seen in planning and estimation activities.		3

GP 3.2.2 Adopt agile leadership styles and adjust the behaviors towards mistakes of people	When a problem or error occurred, team quickly resolves the problem then performs root cause analysis of the problem and takes the actions to prevent reoccurrence of the error.		3
GP 3.2.3 Encourage people in the organization to participate in learning, teaching and improvement	People learn and improve by their own efforts. They are not trained about agile approaches or practices.	A knowledge management system should be established to share learned knowledge and organizational memory covering management activities.	1
GP 3.2.4 Collect measures to support learning and improvement	Management aspect's activities are not followed through measures just direct observation.	Agile metrics are needed to be defined and collected for management activities.	1

CURRICULUM VITAE

PERSONAL INFORMATION

Özden Özcan Top was born in Bolu Turkey in 1982. She received her bachelor degree from Industrial Engineering in Yıldız Technical University in 2005. She received her M.S. degree from Information Systems in Informatics Institute of Middle East Technical University in 2008. Her research interests include agile software development, process assessment, process improvement, software project management and software engineering. You can contact her at ozdentop@gmail.com

WORK EXPERIENCE

Company **:** FUJITSU Technology Solutions, Ankara
Position **:** Quality Specialist, Project Manager
Duration **:** August 2011, March 2014

Company **:** Middle East Technical University, Ankara
Position **:** Research Assistant
Duration **:** November 2007 **-** July 2011

Company **:** InterMedia A.Ş - Ankara
Position **:** Business Development Manager
Duration **:** July 2006- March 2007

EDUCATION

Ph. D Degree (2008-2014)

Information Systems Department /Middle East Technical University

Ms. Degree (2006-2008)

Information Systems Department /Middle East Technical University

Thesis Title: "Functional similarity impact on the relation between functional size and software development effort"

Bachelor's Degree (2000-2005)

Industrial Engineering, Yıldız Technical University

SCIENTIFIC PROJECTS INVOLVED

Title : 113E528 – Agile Maturity Model and Agility Assessment Tool
Duration : April 2014-March 2016
Supporter : The Scientific and Technological Research Council of Turkey (TÜBİTAK)
Role : Researcher

Title : 109E020 - Software Benchmark Dataset for Estimation and a Process Oriented Estimation Method
Duration : September 2009-October 2011
Supporter : The Scientific and Technological Research Council of Turkey (TÜBİTAK)
Role : Researcher

Title : 107E010 – A Unified Effort Estimation Model and Tool Set
Duration : January 2008- January 2010
Supporter : The Scientific and Technological Research Council of Turkey (TÜBİTAK)
Role : Researcher

PUBLICATIONS

Özcan Top, Ö and Demirörs, O. «Assessing Software Agility: An Exploratory Case Study» Software Process Improvement and Capability Determination, Communications in Computer and Information Science Volume 477, 2014, pp 202-213

Özcan Top, Ö and Demirörs, O. «Assessment of Agile Maturity Models: A Multiple Case Study» Software Process Improvement and Capability Determination Communications in Computer and Information Science Volume 349, 2013, pp 130-141.

Özcan Top, Ö Software Agility Reference Model v1.0, Informatics Institute, METU/II-TR-2014-37

Özcan Top, Ö Software Agility Reference Model v2.0, Informatics Institute METU/II-TR-2014-38

Özcan Top, Ö Software Agility Assessment Reference Model v3.0, Informatics Institute METU/II-TR-2014-39

Ö. Özcan Top, "AgilityMod: Software Agility Reference Model v3.0 Application: Case Study Results," Informatics Institute, METU/II-TR-2014-40, 2014.

Yürüm, O., Özcan Top, Ö, Ertuğrul, M., Demirörs, O. «Yazılım Süreç Değerlendirme Araçlarının Karşılaştırılması: Bir Çoklu Durum Çalışması» Ulusal Yazılım Mühendisliği Sempozyumu, 2014, p360-371

Özcan Top, Ö., Demirörs, O. «CMMI ve Çevik Yazılım Geliştirme Yöntemlerinin Birlikte Uygulanabilirliği», Ulusal Yazılım Mühendisliği Sempozyumu 2013

Özcan Top, Ö., Özkan. B, Nabi, M., Demirörs, O. «Internal and External Software Benchmark Repository Utilization for Effort Estimation» Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA) , vol., no., pp.302,307, 3-4 Nov. 2011

Usgurlu, B., Özcan Top Ö., & Demirors O. (2010). «A Clustering Based Functional Similarity Measurement Approach» Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference, no., pp.371,375, 1-3 Sept. 2010

Ungan, E.,& Özcan Top Ö., Özkan B., & Demirörs O. (2010). Evaluation of Reliability Improvements for COSMIC Size Measurement Results, IWSM / MetriKon / Mensura 2010, Germany

Top, O.O.; Demirors, O.; Ozkan, B., "Reliability of COSMIC Functional Size Measurement Results: A Multiple Case Study on Industry Cases," Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on , vol., no., pp.327,334, 27-29 Aug. 2009

Ozcan Top, O., & Demirors, O. & Turetken, O. (2009) "Making Functional Similarity Count for More Reliable Effort Prediction Models", ISCIS, 504-512, Northern Cyprus.

Ungan E., & Demirors, O. & Ozcan Top, O., & Ozkan, B. An «Experimental Study on the Reliability of COSMIC Measurement Results», Software Process and Product Measurement, Lecture Notes in Computer Science Volume 5891, 2009, pp 321-336

Turetken, O., Demirors, O., Ozcan Top, O., & Ozkan, B. (2008). The Effect of Entity Generalization on Software Functional Sizing: A Case Study Product-Focused Software Process Improvement (Vol. 5089/2008, pp. 105-116): Springer Berlin / Heidelberg.

Top, O.O.; Tunalilar, S.; Demirors, O., "Evaluation of the Effect of Functional Similarities on Development Effort," Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference , vol., no., pp.419,426, 3-5 Sept. 2008

Turetken, O., Ozcan Top, O., Ozkan, B., & Demirors, O., (2008). The Impact of Individual Assumptions on Functional Size Measurement, Software Process and Product Measurement, Lecture Notes in Computer Science Volume 5338, 2008, pp 155-169

CERTIFICATES

2010 – COSMIC Software Functional Size Measurer; COSMIC

2011 – Introduction to CMMI for Development v1.3; SEI

2011 - Professional Scrum Master (scrum.org)

2012 – Standard CMMI Appraisal Method for Process Improvement Training v1.3; SEI

2012 - IT Infrastructure Library (ITIL) Foundation Level

INTERESTS

Photography (nature and portrait), music (Jazz listener), sewing, sports (pilates and yoga), and meditation.