ASTUDY ON INTEGRATION OF AGILE WITH STRUCTURED SOFTWARE DEVELOPMENT PROCESSES

A MASTER'S THESIS

in

Software Engineering

Atilim University

by

NURA ABDELMAGID

JANUARY 2017

ASTUDY ON INTEGRATION OF AGILE WITH STRUCTURED SOFTWARE DEVELOPMENT PROCESSES

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

NURA ABDELMAGID

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE/DOCTOR OF PHILOSOPHY

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2017

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ibrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesisfor the degree of Master of Science.

_____

Prof. Dr. Ali Yazici

Head of Department

This is to certify that we have read the thesis Astudy on integration of agile with structured software development processes submitted by Nura Abdelmagid and that in ouropinion it is fully adequate, in scope and quality, as a thesis forthe degree of Master of Science.

_____

Prof. Dr. Alok Mishra

Supervisor

Examining Committee Members

Prof. Dr. Alok Mishra _____

Prof. Dr. Ali Yazici _____

Asst . Prof. Dr. Murat Ozbayoglu _____

..... _____

Date: 26.1.2017

I declareand guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Nura Abdelmagid

Signature:

ÖZ

ÇEVİK YAKLAŞIMIN YAPISAL YAZILIM GELİŞTİRME SÜREÇLERİ İLE ENTEGRASYONU

Nura, Abdelmagid

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof.Dr. Alok Mishra

Ocak 2017, 80 sayfa

Son zamanlarda,çevik metodolojiler (AMs) tüm şirketlerde yazılım geliştirme hayat döngüsü (SDCL) ile iç içe olmuşlardır. Birçok kuruluş yazılım geliştirme hayat döngüsü süreçlerini artırmak ve pazarlama yapmak için zaman yaratmak için çevik metodolojiler (AMs) ve geleneksel metodolojiler kullanmışlardır.Bu çalışma iş alanında ideal çözüm olarak hem çevik metodolojiler (AMs) i hem de geleneksel metodolojileri tanıtmışlardır.İlk olarak, yazılım geliştirme hayat döngüsü boyunca değişiklikler hakkında düşünür, müşteri daha sonra fikrini de değiştirebilir.Bundan dolayı metodolojilerin birleşimine ihtiyaç duyulmaktadır.İkinci olarak, çevik metodolojilerin (AMs) kuruluşlara faydası olabileceği gerçeğine rağmen,karışık bir çerçeve olarak yazılım geliştirme hayat döngüsü (SDCL)'in bazı aşamalarında geleneksel metodolojilere ihtiyaç vardır.

Anahtar kelimeler: Çevik Metodolojiler (AMs) , Geleneksel Metodolojiler, Yazılım Geliştirme Hayat Döngüsü (SDCL), Bileşme Çerçevesi, Çevik Uygulamalar.

To My Parents

ABSTRACT

INTEGRATION OF THE AGILE WITH STRUCTURED SOFTWARE DEVELOPMENT PROCESSES

Nura, Abdelmagid

M.S., Software Engineering Department

Supervisor:Prof.Dr. Alok Mishra

January 2017, 80 pages

Nowadays, Agile methodologies (AMs) have interred SDLC in all companies. Many organizations have used AMs and traditional methods to enhance SDLC processes and make time to market. This study has introduced both methodologies as an optimal solution in the business area. First, thinks about changes during SDLC; also customer may change his mind. So there is a need for the combination methodologies. Second, in spite of the fact that AMs can be beneficial to organizations, there is a need for traditional methodologies in some phases of the SDLC as a mixed framework.

Keywords: Agile Methodologies (AMs), Traditional Methodologies, software development life cycle (SDLC), combination framework, agile practices.

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Assoc. Prof. Dr. Alok Mishra for his guidance and insight throughout the research. Thanks also go to head of the software department Prof. Dr. Ali Yazici. To my husband, I offer sincere thanks for his continuous support and patience during this period.

CONTENTS

ABSTRACT iii

ÖZ iv

ACKNOWLEDGMENTS vi

CONTENTS vii

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF ABBREVIATIONS xii

CHAPTER 1 1

INTRODUCTION 1

1.1.Background and motivation 4

1.2. Problem statement 5

1.3. Thesis Scope 6

1.4. Thesis Structure and Methodology 6

CHAPTER 2 8

2.1. Different types of agile methodologies (AMs) 8

2.1.1. Scrum 8

2.1.2. Extreme Programming (XP) 9

Extreme Programming (XP) 9

2.1.3. Test-driven Development (TDD) 9

2.1.4. Unified process (UP) 10

2.1.5. Kanban 10
2.1.6. Feature-Driven Development (FDD) 10
2.1.7. Crystal Methods: 11
2.1.8. Lean 11
2.2. Benefits of the Agile Methods 11
2.3. Benefits graceful most important are as follows: 12
2.4. Criticism of Agile Development 13
2.5. When to use the Agile Model 13
2.6. Traditional model 14
2.7.1. Waterfall model 14
2.6.2. Incremental Model 15
2.6.3. Agile model 16
2.6.4. Iterative model 16
2.6.5. Spiral model 16
2.6.6. Big Bang Model 16
2.7.1. Why We need and use the combination frameworka 16
CHAPTER 3 18
3.2. Related work 19
3.2.1.Diferent kinds of the traditional methodologies and AM how can be reach succed point in business area 20
3.2.2. Problem can appear while using the Agile development or traditional software development or both as acombination framework 20
3.2.3. Logical Application as a first draft for Physical Application 22
3.2.4. How can organization improve SDLC 23
3.2.5.The reaction of moving from traditional methodologies to LM to AM 24
3.2.6. Success after a good estimation 24
3.2.7.The combination framework and innovation 26
3.2.8. Which method will be integrated to make time to market 27
3.2.9. Organization and moving towards from traditional methodologies to AM. 28
3.2.10. AM toward contract negotiation 29

CHAPTER 4 30

THE EVOLUTION SCENARIO OF THE COMBINATION framework 31

4.1.The relationship between the framework of Agile (AM) and Conventional methodologies and documentations 31

4.2.The new approach journey of the combination framework (traditional methodologies and AM) 32

4.2.1. The combination framework types Table 1 32

4.2.2. The combination framework and methods used in the framework table 2 32

4.2.3. This section interest on present the combination Table 3 33

4.2.4. The traditional methodologies or AM that used according to the authors 33

4.2.5. AM types as reviewed by each author 34

4.3.Discussion 34

CHAPTER 5 41

THE BENEFIT OF USING BOTH AM AND traditional methodologies AS A COMBİNATİON 41

5.1. Conclusıon 41

5.2.Future work 42

REFERENCES 43

APPENDIX A 47

Table A 1.The table 1 is describing The used work and evaluations 47

Table A-2.The described framework combination or compression frame work 53

Table A-3. The combination framework that can be used in project domains. 54

Table A 4. The main differences between traditional methodologies and AM 60

Table A-5. Types of AM and the attributes 65

LIST OF TABLES

TABLE **Page**

Table 3-1.The comparison between Traditional Methodologies and AM 19

Table A 1. The table 1 is describing The used framework and evaluations … 46

Table A-2 The described combination or compression framework …… 52

Table A-3. The combination framework that can be used in project domains…….. 53

Table A 4. The main differences between Traditional Methodologies and AM …... 59

Table A-5. Types of AM and the attributes …… 64

LIST OF FIGURES

FIGURES **Page**

Figure 1-1. Software development life cycle (SDLC) adopted from..... 2
Figure 1-2. Agile software development life cycle adopted from(Awad,2005). 4
Figure 2-1. Scrum processes life cycle adopted from(Glaiel,2012). 9
Figure 2-2. Feature driven development processes adopted from(Awad,2005) 10
Figure 2-3.Waterfall life cycle adopted from Stoica et al., (2013) 15
Figure 2-4. Incremental model diagram (Stoica,2013) 15
Figure 2-5.The combination framework framework..... 17
Figure 4-1. Number of the study papers used framework as comparison and combination..... 33

LIST OF ABBREVIATIONS

The terminology used during the thesis journey

The full term	Abbreviation
Agile Methodologies	AM
Software Development Life Cycle	SDLC
Agile Methodologies	AMs
Agile Software Development	ASD
Large Projects	LPs
Lean Methodology	LM
Test Driven Development	TDD
Capability Maturity Model Integration	CMMI
Extreme Programming	XP
Rational Unified Process	RUP
Software Development Model	SDM
Test First Development	TFD
Dynamic Software Development Method	DSDM
Lean Software Development	LSD

CHAPTER 1

INTRODUCTION

Software products have been a part of human's daily lives for over 50 years now. Software development began as a disorganised effort, commonly known as "code and settle". The product was created without a lot of planning, and the plan of the framework was established from amongst numerous transient choices (Poppendieck and Poppendieck, 2007). This worked well for small framework; yet, as these framework developed, it turned out to be harder to include new components and bugs were harder to settle. This style of software development was utilized for a long time until a new idea was presented as a combination Methodology (Pikkarainen et al., 2009). Systems force a taught procedure upon software development in order to make the SDLC more reliable and proficient. Traditional Methodologies are planned in which work starts with the elicitation and documentation of an entire arrangement of necessities, followed by building and abnormal state outline improvement and assessment. Because of these overwhelming perspectives, this approach became known as SDLC. So there is aneed for the combination framework to enhance the SDLC processes and pike up the optimal solution for any problem that appear during the development phases.As traditional methodologies and AMs complementray each other and can improve the business qualification skilles according to the customer need.The SDLC is summurized in Figure 1-1 below.

Figure 1-1.Software development life cycle (SDLC)adopted from Stoica et al. (2013)

A number of specialists discovered that this approach towards SDLC was rather disappointing when faced with the task of overcoming changes. Accordingly, some decided to form procedures and practices to tackle such unavoidable changes. These efforts depend mainly on iterative improvements, a strategy that was presented in 1975 and which has been regarded as “light-footed” systems. The term "agile" first came into spotlight in 2001, when seventeen process methodologists held a meeting to talk about future patterns concerning software development. They saw that their strategies had numerous attributes in like manner of the combination framework, hence the term “agile” was coined, implying that the strategy is both light and adequate. In the end, the "Agile Alliance” and its application to agile software development (ASD) took place. According to Kuusinen et al. (2016) and Käpyahoand Kauppinen (2015), agile strategies rather emphasise individuals, cooperation, working software, client joint effort, and change as opposed to procedures, devices, contracts and plans. Many of the agile ideas first surfaced in the 1970s. According to Abrahamsson et al. (2010), AM are increasingly becoming

popular in the business in spite of the fact that they may accommodate a blend of both acknowledged and disputable software designing practices Richter et al. (2016) and (Käpyaho and Kauppinen,2015).

The software product business would, in all probability, find that particular venture trademark, for example, objective, scope, prerequisites, assets, engineering, and size will figure out which system suits them best - whether nimble or SDLC, or possibly a cross-breed of the two (Bishop and Deokar,2014). Over the past few years, narrative proof and examples of overcoming adversity by prominent experts is an indication that utilized techniques are more successful and reasonable for certain circumstances. Be that as it may, observational studies are desperately required for assessing the adequacy and the potential outcomes of software development strategies. In today's extending capriciousness and powerlessness, gifted people need towork in such fashion as they have more control over how they function and collaborate with companions, clients, and administration. Pressing issues are changing (Bishop and Deokar, 2014) and so are individuals and their thoughts. While there is still a requirement for plan-driven style improvement and administration, in a number of occasions, the greater development lies in adopting AM and adaptable approach.

This study examines the processes of the combination framework and AM approaches for their reasonableness in software development, and a survey is made of the opinions of specialists to figure out which strategy suitable for the project.AM is a particular approach to increase the role of management in software development. This method assists teams in responding to the unpredictability of constructing software while using incremental, iterative work arrangements that are ordinarily known as "sprints". sets the levels for understanding ASD. The main idea is to understand the problem domain for AM and, then, to provide greater detail within the three dimensions in agile ecosystems: chaotic perspective, collaborative values, and barely sufficient methodologies (Highsmith, 2002). ASD practices may also address the interest for adaptability in light of the commercial aspects of the industry, while careful risk management is still required to guarantee ventures meet their targets.

When it comes to amusement software as a mix of workmanship and innovation, developments teams need to make sure there is a right combination of

aptitudes (Schmalz et al., 2014). Huang et al. (2001) have clarified that due to agile manufacturing, a dynamic partnership is rapidly and dexterously found as indicated by the market varieties and requests. In this way, the suggested procedure plan ought to be made professional rapidly. ASD has altogether affected mechanical software development rehearses. Notwithstanding, in spite of its wide ubiquity, there is increasing concern about the role of software design and its significance in coordinated methodologies. Those in favour of the place of engineering in accomplishing quality objectives for vast programming concentrated framework question the adaptability of any advancement approach that fails to properly consider the design issue. This particularly applies to spaces, for example, vehicles Abrahamsson et al. (2010). In addition ASD processes life cycle can be summarized as cycle of learning and team collaboration as presented in figure 1-2 below.

Figure 1-2.Agile software development life cycle adopted from(Awad,2005).

1.1.Background and motivation

Based on this study, the purpose here is to fill in the gap of methodologies by conducting a detailed review of both SDLC and AM. Regarding the traditional method , this study interduced a number of papers, including Waterfall, Unified Process, and spiral model. This study further discusses an overall view of the characteristics of traditional methodologies.Forthermore, the same procedure is followed for AM; introduced series of agile approaches are introduced, such as Extreme Programming (XP), Scrum, Dynamic System Development, Feature Driven Development, and Adaptive software development to underline the characteristics of agile methods. Furthermore, this study carries out a comparison of the different AM

so as to highlight the similarities and differences between them. The following segment scrutinizes the restrictions of each SDLC and AM.

Organizations presently utilizing the conventional Waterfall method for creating software development may begin by shaping a little group, perhaps 4 or 5 individual developers. The group ought to compile all their abilities to create software products of high quality. Following this, the challenges will be addressed associated with the implementation of combined methodologies in the industry based on the opinion of practitioners as well as anecdotal evidence. To conclude, this study conducts a questionnaire to gather feedback from software developers in Perth, to determine which methodology is most commonly used to develop software alongside their views of AM and traditional methodologies.

In the following, an overview is made the processes of agile (and what has been called "agile venture management ") and in addition a basic meaning of the AM for any beginners starting out in software design and development:

- At a high level, traditional methodologies ventures distribute broad timeframes for Requirements gathering, plan, development, testing and UAT, before at ultimately conveying the project.
- AM ventures have Sprints or cycles which are shorter life cycle (Sprints / emphasess can shift from 2 weeks to 2 months) amid which pre-decided components are produced and conveyed.
- Agile projects can have one or more iterations and deliver the complete product at the end of the final iteration.

1.2.Problem statement

Traditional methodologies are still widely used because they consist of organized and structured processes. The benefits of such methods include delivering working software’s when the requirements are fixed Richter et al. (2016); Kuusinen et al. (2016). Yet, these methods have many shortcomings, such as failure to adapt to the bussenes requirement changes and lack of intensive collaboration with customers by the development team at later stages of the SDLC. Instead, Agile methods are

outstanding since they can be adapted and used in place of the traditional methodologies because they focus on customer involvement and consider them as team members. Feedback can be speeded up and used to improve the development process (Pikkarainen et al., 2009). Therefore, the optimal solution is to use a combination framework (AM and traditional methodologies).

1.3. Thesis Scope

This research introduces the use of traditional methodologies processes and ASD processes, entitled as the "Integration of the Agile with Structured software development Processes". While identifying the different types of these methods, it also introduces their advantages and disadvantages as framework. The use of these frameworks is studies in many different companies and industries (of varying sizes) or according to (CMMI level). The proposal can be helpful in the business domain, and the results are collected based on papers (conferences and journals) and authors to show many types of combination framework used in the practical phases followed in companies and organizations.

1.4.Thesis Structure and Methodology

The first section is a brief introduction to SDLC and traditional methodologies and the combination framework, together with the relationship between AMs and traditional methodologies and why we need both as a set to represent past-to-present developments. In the next section, the main objective of the thesis, problem statement, limitations of the thesis and the structure of the thesis are mentioned alongside an agile definition and its different types. When agile can be used is also mentioned in Chapter One as well as information about how it can be Beneficial's on the main thesis topic. Criticism of agile software development (ASD), when to use AMs, traditional methodologies follows in this chapter while a comparison is made between traditional methodologies and AM related works.

In the next section, some reviews are introduced about the relationship between the combination framework of (AMs and traditional methodologies) and documentation. Next we will take a gander at the development of the approach of a combination framework. the results are introduced as to which type of combination

frameworks for Agile exist. Table 3 illustrates a study as the combination frameworks applicable in project domains.The main differences between traditional methodologies and AM are mentioned in this section, together with the main types of AM and different authors as appears in Table 5.In the next section 4discusses the need for combination frameworks, how they can be beneficial, how many different types of AMs can be joined to other types of traditional methodologies, the purpose of the combined framework, and finally when the combination framework can be used.in the end, the discussion and conclusion are presented.

CHAPTER 2

AGILE SOFTWARE DEVELOPMENT AND TRADTIONAL METHODOLOGY

This chapter will present the need of using the combination framework during the SDLC and how it can be beneficial in some way. Also the literature review will be present in this section.

2.1. Different types of agile methodologies (AMs)

2.1.1. Scrum

Scrum in Rugby occurs when members from each team huddle closely together and clash with the members of the opposite team in an attempt to top down the playing field (Abrahamsson et al., 2010). This involves self-management and self-motivation with effective communication between the scrum master and the stakeholder to collect useful feedback about the short daily meetings (15 minute) in each iteration. (Kumar and Bhatia, 2014). Also, the features that must be implemented are recorded in a list of unsolved orders. The client decides which orders will be implemented in the next sprint, the scrum master is a person who is responsible for solving any active problems and empowering the team members during the sprint (Stoica et al., 2013). Scrum processes life cycle is presented in Figure 2-1 below.

Figure 2-1. Scrum processes life cycle adopted from(Glaiel,2012).

2.1.2. Extreme Programming (XP)

Extreme Programming (XP**)** was defined by Kent Beck, Ward Cunningham, and Ron Jeffries. XP suggests the values of the following aspects:

- Community;
- Simplicity;
- Feedback; and
- Courage.

XP is an easy methodology for small- to medium-size teams working on products that have unclear or shifting requirements (Aitken and Ilango, 2013).

2.1.3. Test-driven Development (TDD)

Is test-driven progress products that rely on the repetition of a short development cycle tool. First, the engineer writing a computerized test that characterizes the change sought or new capabilities (Soundararajan and Arthur, 2009); therefore, they provide the measure of the code base to end this test. Finally, the restructuring of the new law standards worthy. Identified with the first test development. Kent Beck discovery system, said in 2003 that TDD enables basic plans and puts certainty.

2.1.4. Unified process (UP)

Unified process (UP) unified process (UP) is a system for the production of items with the basic properties that are exploited for the event-driven, design-driven, frequent, and the additional, parallel, facing chance-, in protest against the incident, and based on the sector. And it called on the otherwise "the development of standardized software" (USDP) and produced by Grady Booch, James Rum and Ivar Jacobson, known as the "Three Amigos." UP and gives the basis for the implementation of regulation programming construction projects, and the structure is made of breakthroughs and controls the primary and secondary.

2.1.5. Kanban

Kanban has been paid by the Toyota Production System, which is also just in time manufacturing approach.

2.1.6. Feature-Driven Development (FDD)

This agile software development process used for short cycle screen (Aitken and Ilango, 2013), identified by the frequent development. FDD joined the main conditions are favorable methodologies light of other famous feet along with other cognitive accepted industry procedures. Kumar and Bhatia (2014) expand the scope of the tasks effectively and groups much larger on the sizing. Feature driven development processes is presented in Figure 2-2 below.

Figure 2-2.Feature driven development processes adopted from(Awad,2005)

2.1.7. Crystal Methods:

Alistair Cockburn is the writer of the "Crystal Methods". According to Stoica et al. (2013), (related to people and skills) Cockburn is a "procedure prehistorian" who has talked with many venture groups while attempting to separate ‘what really works’ from ‘what individuals say ought to work’. Crystal concentrates on the general population parts of the development cycle as:

- Coordinated efforts;
- Great citizenship;
- Participation; and
- Teamwork and Collaboration.

2.1.8. Lean

The objective of using this methodis to reduce cost and deliver ontime within budget .Also, this kind of AM can improve the team productivity and innovation (Abrahamsson et al., 2010).

2.2. Benefits of the Agile Methods

Agile has grown from the experience with realistic projects by senior professionals in the software of the past. For this reason, the elimination of the challenges and constraints of conventional development, graceful style accepted in the industry as a solution to the best development project (Abrahamsson et al., 2010). Almost every software developer Agile method of one form or another has been used. This method provides a framework to help teams mild (Aitken and Ilango, 2013; Trimble and Webster, 2013). It helps them work better and keep focused on the rapid delivery while enabling organizations to reduce the overall risk associated with software development. Agile method that ensures value is optimized throughout the development process (Yadav, 2015). The use of iterative planning and reactions lead to the teams that can always bring all products that carry the required needs of the client. They easily adapt to changing requirements throughout the process through the measurement and evaluation of project status.

Measurement and evaluation of precise vision and early in the progress of each project a reality. According to Yadav (2015), it can be said that Agile approach helps companies to build just the right product. Instead of trying to market the program before it is written, and graceful style enables teams to improve release during its development, making it a competitive product as much as possible within the market. Maintains the importance of the money market with guaranteed work and the team do not end up as an item is sold in stores. For this reason, the agile development method is an attractive option for stakeholders and developers alike.

There are, of course, those who oppose the view. Some say, this method gives results customers can take to the bank. Although the project may not turn out quite as imagined the client, and will be delivered in time that needs to be produced. Throughout this process, the client and the team may change the requirements for the production of quality required by the client. Customers are pleased with the outcomes, and the team meets the client's needs. Constant change can sometimes give both the client and the team more than he originally imagined the product (Aitken and Ilango, 2013) and (Trimble and Webster, 2013). In essence, graceful style is a winning solution for everyone involved in the field of software development.

2.3. Benefits graceful most important are as follows:

Customer satisfaction through rapid and continuous delivery of useful software Dreesen et al. (2016);

- Focus on people and interaction rather than process and tools;
- Provide a program of work often (every week rather than monthly intervals.
- Encourage face-to-face conversation as the best form of communication.
- Close cooperation between the daily business and developers.
- Continuous attention to technical excellence and good design.
- Regularly adapt to changing circumstances.
- Welcoming the changes until late in the requirements;
- Close partnership with customers.
- The maximum benefit of the listed assets and focus on the most pressing issues, with little documentation.

- The ability to identify and confirm the strategy prior arrangements.
- Only senior programmers capable of making the kind of decisions required during the development process. Thus it has no place for novice programmers, unless combined with resources from the experienced.

2.4. Criticism of Agile Development

As stated earlier, not everyone agrees with the positive reception of the Agile approach; some say:

- It is engineer driven as opposed to client driven;
- Agile spotlights on procedures for getting requirements and creating code and does not concentrate on the product design itself;.
- Also It may not sufficiently address the needs of large organizations and certain types of projects;
- In case of software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the SDLC using this approach;
- It lacks accentuation on essential planning and documentation;
- The venture can without much of a stretch get taken off track if the client agent is not clear about the desired final outcome;
- Only senior programmers are capable of making the kind of decisions required during the development process. Hence it has no place for software development, unless combined with experienced resources.

2.5. When to use the Agile Model

The most ideal and appropriate situations where AM is most applicable are as follows:

- At the point when new changes are should have been executed. The flexibility coordinated provides for change is essential since such new changes can be executed at almost no cost due to the recurrence of new additions delivered;
- To implement a new feature, the developers will not have to lose, say a month's or extended period's work, but only that of a few days, or even only hours, to roll back and implement changes;

- Unlike the Waterfall approach, in AM extremely restricted arranging is required to begin with the venture. AMs expect that the end clients' needs are ever-changing in a dynamic business and IT world. Changes can be talked about and components can be recently affected or expelled in light of input.In essence, this gives the customer the finished system they want or need Abrahamsson et al. (2010).
- Both system developers and stakeholders alike notice that they also can spare more time and enjoy more alternatives than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data, or even entire hosting programs, are available which implies that the project can continue to move forward without fear of reaching a sudden standstill.

2.6. Traditional model

These methodologies are based on the sequential phases, such as requirements analysis and planning, identifying needs, and designing product architecture, testing and deployment Stoica et al. (2013). Some models that are considered traditional nature are: a waterfall, incremental, rapid application development (rapid development) applications, Agile, Iterative, Spiral, The Big Bang, and models. Each of these models has its advantages and disadvantages, and intended for use with a particular goal (Aitken, 2014).

2.7.1. Waterfall model

It is a sequential model, consisting of sustainable development stages that begin with the collection of stage and end with the operating and maintenance requirements phase. In this model, it must be refined and outputs of each phase before the next phase can begin Delworth et al. (2012). The waterfall life cycle sumurized in Figure below.

Figure 2-3.Waterfall life cycle adopted from Stoica et al. (2013).

2.6.2. Incremental Model

This model partitions a system into a set of increments (otherwise known as "multi-waterfall cycles"). A working version is introduced in each increment. Certain features are added to the subsequent increment until the last version is completed. There may be portioning problems that can occur because the requirements cannot be detected early enough Stoica et al. (2013).

Figure 2-4.Incremental model diagram Stoica et al. (2013).

2.6.3. Agile model

This is where software is developed in incremental and rapid cycles. Agile is used in developing software systems where time is critical. The approach requires professional developers to be engaged in the process (Kakar, 2014). An important aspect of the Agile software development strategies is the adaptive software development processes employed Abrahamsson et al. (2010).

2.6.4. Iterative model

In this approach, the process begins with a basic execution of a subset of the software requirements, iteratively upgraded within the upcoming versions until the full and satisfactory one is formed. In each iteration process, plan adjustments and certain features are to be included.

2.6.5. Spiral model

This model integrates certain components of both plan and prototyping-in-stages. It is similar to the incremental model but stresses out more on risk analysis, a process that demands more expertise among the team members Feng et al. (2016).

2.6.6. Big Bang Model

The Big Bang Model is an SDLC where no formal kind of development is followed and almost no planning is required. Indeed, even the customer is not certain about what he/she precisely needs and the pre-requisites are actualized on the fly and with barely any analysis at all.

2.7. Whywe need and use the combination framework

By its very nature, Agile is a phased approach, allowing for many of the pledges to be fulfilled in the early (Aitken and Ilango, 2013) Stoica et al. (2013). Also, there is a high degree of participation clients (Käpyaho and Coppinn, 2015). In traditional methodologies, software is required specifications at the beginning of the project since then, at a later stage, and repair work is usually costly. So, back to the stage of the condition it is not easy and desirable task because it usually requires a reformulation of the operations of the product, calling for more time and expenses Stoica et al. (2013).

Graceful style of the traditional ones because - Because it gradually as mentioned before - it does not need to complete specifications at the beginning of the development process (Yadav, 2015) varies. However, flexible methods limit the usefulness of the organizations on a large scale and certain types of projects, requiring senior programmers that can make the right decisions necessary during development processes. As a result, it can be concluded that the benefits of flexible methods can be better when they are in combination with traditional methods of sustainable development. In what follows, we have a detailed look at the different models available for sustainable development. The combination framework sumurized in Figure 2-5 below which contain AM and traditional methodologies.

Figure 2-5. The combination framework

CHAPTER 3

THE DIFFRENCE BETWEEN TRADITIONAL METHODOLOGIES AND AM IN MANY PHASES

3.1. The comparison between traditional methodologies and AM

Traditional approaches are mainly guided by organized processes with due and enhanced consideration of the shifts in the sources according to the measures as well as enhancement of the processes within each lifecycle. Each process (phase) has it own defined outcomes; for example, the outcome of the requirements analysis phase is the requirements specification document. Communication among team members and stakeholders are formally organized, and stakeholders have an important role during the early stages of the development process; especially throughout the specification process. This role is less crucial as the later processes and stages come up. The life cycle model specifies the tasks to be performed and the desired outcomes of each phase, which means the work is done through a complete list of details.

in contrast to this approach, AM depend on people and their creativity rather than on processes. Development will be done through short iterative cycles and effected by software services, and effective communications which includes negotiations (rapid feedback), thereby speeding up the decision-making process. The role of the stakeholders' is important in all of the repeated processes. This calls for the development team to work with these stakeholders in a collaborative style to draft out a detailed list of tasks planning’s to be completed in the next iterations.

Table 3-1. The comparison between traditional methodologies and AM

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

3.2. Related work

In software development processes, there is a need for new approaches to enhance the development processes. With this in perspective, and according to Richter et al. (2016) and Soundararajan and Arthur (2009), a soft-structured framework can be made as a combination of agile principles and traditional methodologies to address the main issues regarding adaptation to change associated with large-scale organizations and systems. This framework is a combination of two parts of software structured requirements with an agile mindset, leading to the application of the AM model and traditional development processes which can be suitable for both small- and large-scale systems.

Agile requirement engineering is still a generally new field of research and the developing utilization of agile strategies in large projects is driving organizations to search for more formal practices for requirements engineering (Käpyaho and Kauppinen, 2015). Two other options have been offered as to the implementation of system functions: fully designed and validation of the applicability of agile requirements generation model.

Over the last decades, the main issue surrounding software engineering is that of producing high quality software products delivered within budget and on-time (related to early delivery and cost reduction). Furthermore, upon payment against the costs, the customer has to do so with utmost content and pleasure with the product they receive (related to customer satisfaction and validation).

3.2.1. Diferent kinds of the traditional methodologies and AM how can be reach succed point in business area

Software development duration is a loop of phases that can enhance the software product quality by using some of the effective methods; though there are challenges when using a combination of traditional models such as waterfall, iteration, spiral, etc., and AM such as scrum, feature-driven development (FDD, LSD, Kanban, crystal groups and so on, such merging of the methods is done in order to gain the benefit of using both styles of traditional and modern methodologies. Obviously, the final combination of this framework upon actual and practical project implementations demanding such a mixed approach can have advantages and disadvantages as well (Kumar and Bhatia, 2014). Currently, most software companies have adopted combination framework practices in their development processes in order to guarantee 100% success in the final product.

When tailored in the right way, Agile can be seen as an optimal solution for adaptation in ng in large companies in terms of different factors such as culture, business domain, project, and project concerns such as size, team distribution, business model, and stability. this model is used for specific organizational contexts, and the main reason for using agile is that it can be usually adopted in a bottom-up form by practitioners not necessarily within the team that has been rounded up for the purpose of software development collocate. In this way, other areas often start working towards with the conventional practices declared by Kuusinen et al. (2016) using online surveys and interactive workshops.

3.2.2. Problem can appear while using the Agile development or traditional software development or both as acombination framework

A typical and conventional organization is characterized as conservatively composed, with particular jobs, different offices, multi-levelled lines of feedback and product stories, all the way to the team leader, decision making, and top-to-bottom

arrangements (Richter et al., 2016). Richter et al. (2016) state that after their exploratory semi-structured interview study, they notice that AM can be used for various types of software development processes, even for distributed and complex projects, leading to a mixture of traditional and agile-centred companies as far as software development is concerned. More specifically, the point of their exploratory and semi-structured interview working study was to clarify the issue of using agile software development processes in Germany within plan-driven organizations, leading to the detection of many problems that can appear while using ASD. A final conclusion from that study was that it is hard to integrate Agile concepts into software engineering projects in a plan-driven organization. Such projects could not reach a successful point within the rankings of typically amd conventionally organized corporations, without proper leadership and knowledge of agile-centred managerial skills.

Another kind of problem which can appear while using the Agile development approach in software companies is the issue fragmented documentation. Agile projects documentation is more dispersed than traditional projects, bringing one to the optimal solution that teams themselves should document who is working on which projects, what kind of work unit and where and how to contact him for the project (related to light-weight documentation and user story). On top of this, the information has to be digitized and easily available. This study strongly believes that every single piece that might be beneficial for future projects, too, has to be recorded and stored in a standard and proper manner.

According to Kumar and Bhatia (2014), the main issues of the spiral model big shift from agile to spiral is that each phase is completed with a product story and can be taken as feedback by both the designer and the programmer; also, an extended risk analysis can be made than s to this model, which is often deployed for large-scale projects. The spiral, however, does not work well with small projects and can provide flexibility within the work environment.

The Dual Application Model (Aitken, 2014) includes the development of an intelligent programming application aimed to encompass useful prerequisites alongside a physical programming application concentrated on changing sensible applications to compensate for the non-useful prerequisites. It may have some favourable charactersitics, though, such as drawbacks regarding alternative ways to develop a software, but meets two of the standards proposed for Agile software

engineering. The systems and devices have been suggested to bolster the Dual Application Model. Though, these are not core features of the approach, which offers a whole practical choice as to characterizing and indicating (in code) the space planning of the software. Specialists depend on these features to outline the specifications and execute the product so as to meet the non-functional pre-requisites.

It Dual application includes the developments of two software applications rather than the traditional deliverable one. The principal application is mainly the "Logical Application" and is runnable sensible model of the space solution/software issue. It is autonomous of any arrangement innovation and the non-functional prerequisites. This application is created with the goal for quick formation and effortless updating and renewal properties. The Logical application is shaped with partial feedback from the user, item proprietor, and additionally space expert, the second application is the Physical Application also, it is a physical model of the product arrangement and what is in the end conveyed as the product arrangement.

3.2.3. Logical Application as a first draft for Physical Application

The Logical Application is a first draft for Physical Application; however, the application required augmenting and alteration to keep running on the physical execution stage and meet all the non-practical necessities of the product solution such as persistence, ease-of-utilization, speed, and dependability. The development of the Physical Application depends on the space and mastery of the software designers, despite the fact that the item proprietor guides the application through constant demand for different non-utilitarian properties (Aitken, 2014).

A new framework which is a result of combining AM with traditional methodology can be helpful to understand the relationship between traditional software engineering and agility. Jiang and Eberlein (2008) have proposed a novel, five-dimensional way to emphasize the main points throughout the historical and technological contribution of the methodologies, namely: ontextual analysis, historical analysis, analysis by analogy, phenomenological analysis, and linguistic analysis. Tthe attributes are different in terms of both methodologies, qualification skills, and the income level for those engaged in the process, the latter naturally as a result of the profit made from the software developed. Furthermore, one should not

fail to consider the organizational and technological state and the the business environment to effectively execute them operations.

3.2.4. How can organization improve SDLC

Jiang and Eberlein (2008) outline the best practices of the traditional software development processes and AM from industries; the output of this framework is the main relationship between both, and one of the key debates between supporters of traditional methodologies and AM appear in the available framework advocated by the waterfall model and other plan-based processes models supported by the CMMI to mitigate the main issues within this framework calls for the utilization of high-level practices from both XP and the CMMI to facilitate the engineering process. In fact, according to Jiang and Eberlein (2008), “agile practices support eleven components of CMMI”, while the organisation level visible in CMMI complements that of Agile. In practice, the software community mostly understands the advantages and disadvantages of these models as well as and how to use the combined methodologies to improve them in software processes, as opposed to focusing on improving one methodology alone. Thus, suppliers need to make background checks by looking at the origin of each methodology and the social environment supporting the versions of this framework (helps if you establish the relationship between social environment and software methodologies.

Based on the research by Bishop and Deokar (2014), in their exploratory study, there are variables contributing to the sense of inclination to adopt an agile approachin the industry. The underlying investigation swerves around the so-called ‘Five Factor Model’ of identity, establishing that elements give only partial clarification as to why some intend to prefer the agile approach. These methodologies have wide and different orientations: while AM take an iterative approach, the Big Bang approach is somehow a traditional one. (wrong place to compare agile and big bang At least, re-organise paragraphs to have smooth progression of ideas. Interestingly enough, identity does not fully account for an individual's inclination for agile strategies. On the off chance that identity totally clarified inclination than the ability to alter preference would require changing identity. Since identity is not a restrictive figure, this leaves an open door for recognizing extra calculates of impact inclination. Once those extra components are recognized, then projects can be intended to control inclination utilizing the additional factors (Bishop and Deokar, 2014). As a result of the integrated

framework of traditional methodologies with AM, customer value will be increased parallel to the overall team satisfaction during the whole SDLC (Aitken and Ilango, 2013).

3.2.5. The reaction of moving from traditional methodologies to LM to AM

Trimble and Webster (2013), in their study to journey from traditional methodologies to LM to AM, detailed their experiences and applied an amplified learning approach in the process of engineering a software for early delivery of a turn out control user application structure. According to their work, the impact of the shift on the suppliers' team, customer relations, software product quality assurance and cost can affect any company or organization budget (regardless of their size). That is why the stakeholders decide to invent such framework to move from traditional methodologies to LM and then to AM, discovering along the way that while using LM, short delivery life cycles are more easily achieved (related to early delivery and technical excellent). Thanks to such short delivery time, they can identify and make room for frequent customer integration while testing each deliverable software product can be done in yet even earlier stages.

3.2.6. Success after a good estimation

Put in a different way by Palmquist et al. (2013), 'traditional World/Agile World discourses' are the way to achieve a similar goal – that is, deliver a quality software product in an anticipated, effective and responsive way. This is all despite the fact that the customer usually wanted to know about the software product and when it can be delivered and how long the product requirements take to implement (Trimble and Webster, 2013). In general, the traditional methodology for software has been successful in estimating code lines (cost and time), thereby making it a part of the common practice to develop each software product as assigned by contract between a team and the customer. In short, the traditional estimations are used to calculate the anticipated end date of the project and the iterations during the project upon close collaboration with the customer, who now has the opportunity to monitor the project progress. In moving from traditional methodologies to LM to an AM, experiments reveal that the following framework steps can yield better results:

- Take light-weight documentations into consideration side by side the high-level requirements priorities (related to light-weight documentations and cost reduction);
- Monitor project progress to allow the team members and the customer to know everything about the software product after upon signing a contract charactersied by transparency (related to customer value and team member's satisfactions and self-management) (also related to validation and veryfication);
- Get immediate feedback in each iteration.

During the software development life cycle (SDLC), sometimes there will be a risk in project management affecting those relied upon for prototyping. Pre-production, decisions and applied AMs can prevent such risks. Managing risks is also known as 'risk mitigation' (related to risk mitigation and risk reduction). Schmalz (2014) state that cost can be reduced simply by implementing formal risk management to informal practices. No matter the amount of vigor and enthusiasm put into a work by software designers, general regard for formal risk management is low. Yet, informal practices (for example prototyping, AMs, and pre-production choices) are the main point of such risk management here. Custom-designed software requests as specific items in the market are the main objective to be achived as opposed to creating or actualizing framework for in-house or ordinary and non-specific products. Obviously, conclusions cannot be drawn from such a constrained example. However, the outcomes here do point to the requirement for further examination concerning how best to match extended management and risk management philosophies to the sort of venture work being embraced. The unpredictability of the simulation software industry may require casual venture management efforts while keeping in mind the end goal is quick and adaptable adjustments and reactions to the market.

In any way, different elements that appear throughout this planning may show the extent to which software laboratories can concentrate on building extended SDLC. These incorporate the requirement for project management and specialized preparations for advanced personnel within the team and acknowledgment of that, while coordinated development practices may address the interest for adaptability due to the commercial nature of such activities, cautious risk management is still required to guarantee projects to meet their ultimate targets.

3.2.7. The combination framework and innovation

At last as a mixture of framework and innovation proposes the requirement for watchful constitution of development groups to guarantee the right adjustment of abilities. Most agile process practices are adjustments of those same practices that have been touted by methodologists throughout the past two decades and that can be found in more thorough "traditional" procedures. This has been perceived by coordinated process advocates who advocate the idea that the distinctions lie not in the individual practices, but rather in the way they are assembled. The cobbling together of best practices to make forms that fit an improved version's qualities and development objectives has been upheld by various methodologists and has brought about no less than one tailorable process framework.

With respect to the adaption of AM (related to adaptive to change) over conventional approach of SDLC, this is a demonstration of all intents and purposes valuable to mechanical association. The work by Yadav (2015) incorporates a similar investigation of a dexterous and iterative approach to software development There are a few points that separate agile approaches from Iterative as follows:

Incremental development: Agile is Iterative in addition to the inclusion of preferences. Yet, an iterative approach is still an arrangement of mini waterfalls. In spite of the fact that emphasis is adaptable in principle, in common sense versatility can bring about distress in processes. Agile is both iterative and incremental, thereby increasing the chances that a venture can be developed. According to Yadav (2015) customer involvement (related to customer involvement) includes clients and end-customer delegates all through the whole term of the project lifecycle. That will bring about an element which will absolutely fulfill the need for adaptability by clients. Iterative development does not address the absence of business and end-client inclusion as a rule found in waterfall projects. In contrast to this view, an agile approach turns out to be more beneficial when contrasted with iterative approach. Different overviews demonstrate the significance of adjustment of deft processes by means of increment in the number of associations adjusting to such deft procedures because of different agile benefits like increment profitability, cost reduction, improved quality, and the capacity to oversee evolving needs, sustainable development and diminish so as to convey item and so forth.

3.2.8. Which method will be integrated to make time to market.

Despite the fact that different troubles and difficulties are confronted by different associations with respect to accommodating AM, its advantages continue to draw the attention of different firms. Both iterative and AM turn out to be better when compared to traditional methodologies of software development. Yet, the agile approach fulfills a few extra needs over iterative development, thus making it a superior technique for software development in the overall sense of the practice. According to Käpyaho and Kauppinen (2015) and their case study in the software development area, there is a variety of different quick and cheap-to-implement approaches to create prototypes even in the earliest phases of the development cycle. The utilization of prototypes to attain the prerequisites fits the agile thinking in a promising way

Building the" unique" can be viewed as an effective method for achieving agreement on what a team is doing. As Windsor and Storrs state, we require a prototyping of the advancing outline that is understandable to clients and is in line with the natural tendencies of human beings as much as it can be expected. It is immensely challenging to imagine the conduct of an intelligent framework with composed, regularly protracted and complex particulars. Due to the regular propensity of people to comprehend complex framework better as a perception, prototyping can be viewed as a decent method for coming to a shared comprehension.

According to Dreesen et al. (2016) based on their literature review, the combination framework and the communication between the developers and the customer is important. By means of applying AM, the management of communication practices can be made better as a factor to improve the main issues in ASD projects themselves and focus further on the project requirements. If the organization starts with an incremental model, the requirements can be classified into subsets. The model includes various improvement cycles, which make the life cycle resemble a "different waterfall" demonstration. The cycles are again isolated into even smaller cycles and modules that are not as vital to be overseen. Every module conveys requirement design, analysis, implementation and testing. Amid the principal module, a working adaptation of the product is made. Every after rendition includes new components and functionalities to the one before. In this way, when there is a need for adaptation, the organization may quickly move to use its agile approach so

as to increase adaptability to shifting business environments and, eventually, expand its profit margins.

By means of applying agile principles, the company or organization demonstrates its proficiency in re-configuring software development processes that pick up AM to improve skills, computerize the records of choices and elements, and permit business approaches to be moved to a marketplace. This can bring about better substantiated choices and, on top of that, expanded association capacity to reply to market changes and opportunities. Accomplishing ASD requires information regarding how software assurance can be expanded Stoica et al. (2013).

3.2.9. Organization and moving towards from traditional methodologies to AM.

Many organizations move from traditional methodologies to agile ones because the costs are simply higher when using traditional methodologies. In line with this idea, Sumrell (2007) states that during the testing, a cycle can be separated into smaller components for practical, framework and relapse testing. One stage cannot begin until the one before is completed. Testing groups in all organizations remain as a genuine and invaluable resource for those establishments as any stakeholder wishes for their products to be developed by such teams of experts that do not use typical costly, off the shelf automated testing software.

As the company or the organization stakeholders change their views as to software development in the course of time and into an agile approach, and making use of Scrum sprints, the firms may find it difficult to properly migrate from the conventional mechanized testing techniques commonly used by the industry. Such a change will, at any rate, take place in order to reduce cost and deliver the software product early. Related to these issues, Aitken and Ilango, 2013) have pointed out to the the fundamental contrasts and any conceivable inconsistencies between these two software development ideal models. Their work explores various conventional and agile philosophies, strategies, and methods. According to them, the fundamental contrasts between traditional software building and ASD are observed not to be - as one may first speculate upon a quick thought - identified as emphasis on the length or management of the project, but rather as different qualities like the purpose of the models utilized, the motivation behind the models, and the way to deal with demonstrating. LSD does not include stages the way conventional software

development methodologies do, for example, waterfall or RUP. Instead, it is a ceaseless moving cycle of making, organizing, executing, and testing client stories. With this in perspective, quality assurance experts working on LSD span ought to offer their best of administrative skills to improve the item without piling up and behind the development process as a whole or allowing user stories to amass in the 'ready to test' state. To do this, the team should have the capacity to grasp the "pull" attitude of LSD and turn into a fundamental part of the development cadence of the large teams.

3.2.10. AM toward contract negotiation

Given the opportunity, anyone involved in development processes will find out for themselves exactly how fun and successful building important software can be. At last, however, the two methodologies are not seen to be incompatible, prompting to the future plausibility of an Agile Software Engineering (ASE). Customer cooperation over contract negotiation is paramount (related to crystal clear contract review) during SDLC. Obviously, such a presumption can be quite deceitful too that is to say, to expect that TSE (traditional software engineering) would without, hesitation and no difficulty at all, LM toward contract negotiation and joint efforts with the customer (related to customer satisfaction). The reality of the matter is that agreement transaction is basic under traditional software engineering; however, it is reasonable to claim that it is normal under ASD, too. The majority of the business world is not prepared for "time and materials" software development. Once speaking of ASD, two items stand out:

- That they (they being the traditional v. Agile) cannot ensure culmination of all prerequisites, and
- That they certification to add esteem very quickly and to keep doing that (less theagreement be ended) .As a general rule, numerous test software engineering

projects have been contracted in this way too, with organized conveyance (with prioritisation of the requirements in advance) of high need necessities firstly), and other terms and conditions in case if each stage is not completed upon unanimous agreement of all team members out-provisions if stages were not finished agreeably. Then, again test software engineering utilizes contractual transactions to shield engineers from extension crawl, while ASD deals with that issue by never consenting

to convey everything. Yet, to convey such an idea constantly implies that software development companies should start to use the two methodologies as a combined framework to ensure the quality of software product and time-to-market, while achieving maximum customer value (Aitken and Ilango, 2013).

In fact, when speaking of iteration cycles and their durations, traditional software engineering and ASD as best rehearsed are comparative in their quest for short iterations. They both need to utilize every one of the cycles within the process, no matter there might be the possibility of casual and documentation-free ASD. Both look for critical customer contribution in spite of the fact that, when and where required by ASD, the traditional version has for the most part looked for such contributions (and both approaches often stridfe to accomplish it). Also, both create models (otherwise known as ‘documentation’ and related to light-weight documentation) for correspondence among the individuals involved in the process of development. Finally, both expect to convey esteem constantly; however, the nature of this esteem may not necessarily be the same.

CHAPTER 4

THE EVOLUTION SCENARIO OF THE COMBINATION FRAMEWORK

It can be seen to combine framework striker scenario in this episode all over the papers that have been collected and provide the advantage of using framework of the striker in various projects and business areas type. This study introduce the relationship between the striker and the combination of documents, and also the use of such striker in several stages during the SDLC.

4.1. The relationship between the framework of Agile (AM) andConventional methodologies and documentations

According to Käpyaho and Coppinn (2015), can repair the difficulties faced by agile software development models, while also helping to address a number of difficulties related to the development of harmonized software. For example, the lack of documentation and inspiration for the work of the engineering requirement and low-quality correspondence. Similarly, there is a need for mutual practices to maximize the full capacity and potential. TDD regard to compose tests as a necessity or design work, where one can finals functions required by the development.

Before starting the coding phase, it can team members postpone operations for a period of time to write the technical part, which is a sample of about one or two pages dealing with implementation requirements. In essence, this part defines the trade-offs and classifications, such as why the specific design phase has been

selected (Trimble Webster, 2013). It also contains the use cases and test scenarios. Here, team members build a test plan, according to a story from the user and the documents of the Organization, and extract documents with the engineering team to make a formal official sample of the operations of the program to cover aspects such as the concept Maturity Model (CMMI) integration. Yadav, (2015) in the case of a comparative study in the small unit to take advantage of the programs and documents are the primary driver in the morning, while Rob, will benefit from the documents and delivered. According to Stoica et al. (2013), while using the waterfall model, the documents are a plus during the SDLC when new members coming along team. According to Tumbas and Matkovic (2006), the groundwork for progress in this project is to re-use the software level. Measured progress of the projects AM in creating striker data by the amount of software optimized solutions that are used, and not in the light of the period of the software development of the project, the amount of composition law, the amount of documentation provided, and can similarly be used in other new projects

4.2.The new approach journey of the combination framework (traditional methodologies and AM)

4.2.1.The combination framework types Table 1

This section Over Center on the review of the composition framework and any type of use methodologies that can be viewed in table 1 uses a striker and assessments as evidenced by the book and the actual striker as an optimal solution and any type of AM used in this striker and also a kind traditional methodologies also the benefits and disadvantages of the attacker used.

4.2.2. The combination framework and methods that used in the framework Table 2.

This section Intense focus on the review of the composition framework of striker and any type of use methodologies that can be displayed in Table 2 used striker and authors. framework optimal solution as the actual and any type of AM is used in the striker and also a kind of traditional methodologies. Also this kind of striker such as pressure or a group within the organization .Number of the study

below papers use framework as comparison and combination framework summarized in Figure 4-1 below.

Figure 4-1. Number of the study papers used framework as comparison and combination

4.2.3. This section interest on present the combination Table 3

This section or currently combination or the compression framework and name striker combination that can be displayed in Table 3 and used striker and authors. This section also presents a striker and where it can be used in the field of business methodology and practices that are used according to the authors. This section can be useful for the organization or companies that interest on this combination striker.

4.2.4. The traditional methodologies or AM that used according to the authors

according to the authors used, This section interest on present the traditional methodologies or AM. Moerover to the name of the attributes which can be associated based on the methodologies and the side effect of using such kind of methodologies. Byside to these effect these methodologies can be described from

different kind of domains such as (Architecture domain, project domain, people domain (team, stakeholder, customer, development team(suppliers)) …etc.) which can be seen according to table 4.

4.2.5. AM types as reviewed by each author

This division will illustration each type of AM with their characteristic according to the authors.the using of business area and the the benfit each kind of AM was the main function of this table. so table 5 present the AM types which descried by the authors.

4.3. Discussion

In the course of the most recent decade, there have been various changes in the combination framework structure; AM technique and traditional methodologies procedures keep advancing to suit new patterns and above all else to keep organizations aggressive. Nowadays software companies interest in regarding approaches that can be successful "waterfall strategies were supplanted by iterative techniques, for example, spiral model and unified process. According to Pikkarainen et al. (2009) as of now the AM techniques are more worked on sitting tight for the up and coming stages of development. In the course of the most recent years there have been different kind of the combination framework rehearses distributed in papers i.e. gatherings and conferences. they include the combination framework as a Combination of AM with P (building acceptance test driven development) which described by (Käpyaho and Kauppinen, 2015), so ASD as a methodology and P to gain the benefit of using this combination framework such as P is most beneficial for agile projects with some complementary practices. The author such as (Käpyaho and Kauppinen, 2015) has reviewed the Communication and Documentation practices to improve the communications problems and solve the prototyping documentations problems. Actually the framework that as result of combining AM and prototyping can be used in some various of project domains such as LPs, complex software project development, agile projects, Short life cycle projects.

In (2014) the use of Water- scrum -Fall is reviewed by Bannink.while Soundararajan and Arthur (2009) declared that the use of AM and waterfall model can be beneficial throughout focusing on the requirements of each methodology.While Soundararajan and Arthur (2009) and Stoica et al. (2013) and Trimble and Webster, (2013) disscussed that the combination framework is more beneficial and the Customer involvement can be so High (throughout monitoring the

project processes) in AM also is low in traditional methodologies. Other AM practices such as early delivery is reviewed by Sumrell (2007) and Trimble and Webster (2013), Trimble and Webster review early delivery by Moving from traditional methodologies to LM to AM according to the business need and can be used in different kind of projects such as User centric software platform, Projects require short delivery life cycle.

Soundararajan and Arthur soft-structured requirement and tailored development is more practical oriented and more focused on issues of Large-scale complex system, E-commerce company, Small scale system compared to Käpyaho and Kauppinen's framework to Trimble and Webster's framework to Kumar and Bhatia's framework is that Käpyaho and Kauppinen framework is more focusing on Short-term projects; additionally, he points out that the decision should be taken by constant and Builds motivation and Builds acceptance test driven development. While Käpyaho and Kauppinen have gone a step ahead in Solves some issues and using AM with P and building acceptance test driven development, so Leadership according to Käpyaho and Kauppinen in AM is Good leadership and in traditional methodologies is present as Command and control.

While Kumar and Bhatia (2014) have discussed the comparative analysis of traditional methodologies and AM which AM include XP, scrum, FDD and traditional methodologies include W, spiral model, iteration, V-Model, RUP and more focusing on Light-weight documentation and reduce cost.while Kumar and Bhatia are more enhancing reusable component to reduce cost. İn fact Kuusinen et al. (2016) gives Combination framework which is result from traditional methodologies with AM, the concepts can be identified as a summary of AM to implement decomposition tasks or functions. Kumar and Bhatia are more focusing on Decision support system the most important team select SDM that best suit the project while Kuusinen et al. (2016) is more focusing in general practices of the combination framework. The practices remain the same, only that Aitken (2014) focusing on traditional methodologies practices

There is also quantifying agile practices, Communication, documentation practices that makes the good quality items reach Faster time-to-market; by combining waterfall and scrum and early feedback Bannink (2014) discuss this practices. The agile practices and communication of measuring and managing is also reviewed; this includes the team queuing theory and exact items measurements.

Authors as Bannink (2014) and Aitken and Ilango (2013) and Yadav (2015) reviewed this aspect. Customer involvement is also reviewed by Stoica et al. (2013) and Trimble and Webster (2013) and Soundararajan and Arthur (2009). Other agile practices include customer value; TFD; validation and veryfication; reduce cost; rewards; Hide individual performance; Increased performance Kuusinen et al. (2016); adoptive to change; peer programming; Continuous integration (Tumbas and Matković, 2006).

Tumbas and Matković (2006) discuses that the combination framework whether traditional methodologies or AM where software development is incremental (little software releases, with fast cycles), agreeable (customers and suppliers cooperating with close communication) and adoptive (the strategy itself is anything easy to learn and adoptive to change , very much recorded) and adoptive (ready to roll out last minute improvements). Also Tumbas and Matković (2006) presented ASD twelve principles such as team satisfaction; frequent delivery; planning and engineering; individual motivation; usability; Constant orientation ; simplicity; architecture requirements; realization of better efficiency; sustainable development.

Trimble and Webster (2013) also discussed the moving from traditional methodologies to LM to AM that addressed User centric software platform; Projects require short delivery life cycle. Soundararajan and Arthur (2009) reviewed the soft-structured requirement and tailored development which requires to Large-scale complex system; E-commerce company; Small scale system. Sumrell (2007) discussed Traditional automated testing strategy to scrum framework. Kumar and Bhatia (2014) reviewed Comparative analysis of traditional methodologies and AM which can be AM used to improve projects by ensuring short time cycle SQA; spiral model used in government projects in many companies; rapid application development for small projects; spiral model can be used in large and mission-critical projects. Richter et al. (2016) discussed Apply AM in organized corporation traditional methodologies as a semi-structured framework. Gregoy (2016) reviewed the Combination framework as AM and traditional methodologies also reviewed that scrum of scrums for large distribution projects and R- scrum for regulated environments and specific organizational contexts; in large organizations and projects with rapid change. AM leads to implementing decomposition works; The

organization can apply bottom-up change; Tailors agile to suit a hierarchical organization ; Increases productivity and team performance.

Yadav (2015) discussed the Comparative between AM and traditional methodologies which is presented that AM increases the chances to make the project successful; monitors all the processes during each iteration to allow the customer to know everything about the software product; AM includes iteration and incremental; can fix bugs quickly. Aitken (2014) discussed framework to support Dual application model (the best of AM with the best of traditional methodologies) as a combination framework (AM and traditional methodologies) with traditional methodologies practices. Stoica et al. (2013) reviewed the framework of combining AM and traditional methodologies with AM practices such as Continuous improvement; team self-organization; Self- organization team. Aitken and Ilango (2013) reviewed a Comparative between AM and traditional methodologies and announced in AM Cross-functional team members having experience with such problems by using the documentations (problem story, product story, user story). Aitken and Ilango (2013) also discussed the Communication in traditional methodologies is Formal and in AM is informal.

Only Tumbas and Matković (2006) reviewed DSAM is only one of the compared methodologies, that support all the SDLC activities. Tumbas and Matković (2006) reviewed the comperation between traditional methodologies and AM, AM such as XP; scrum; FDD; DSDM; adaptive software development; LSD; crystal family; AM practices.

Stoica et al. (2013) and Kumar and Bhatia (2014) and Aitken and Ilango (2013) and Tumbas and Matković(2006) reviewed the main objective of using XP and how can this kind of methodology can be powerful during SDLC.also they disscused the A programming-focused software development method expand software quality and responsiveness to modified customer requirements by means of simplest coding solutions. An easy methodology for small to medium teams that develop software products with vague or changing requirements**.** Inspired by RUP, XP has six stages and addresses the value of community, simplicity, feedback and courage. XP involves some practices such as planning, TDD, customer involvement, refactoring, pair programming, and continuous integration.Also Stoica et al., (2013),Sumrell (2007),Bannink (2014),Kumar and Bhatia (2014),Trimble and Webster (2013),Aitken. and Ilango (2013),and Tumbas and Matković(2006) has

reviewed the main points of using scrum as one of the AMs to present the advantages of this kind of methology and show that self-management and self-motivation in teams with effective communication between the scrum master and the stakeholder to receive useful feedback upon short daily meetings that last 15 minute.

Tumbas and Matković (2006) and Aitken and Ilango(2013) and Stoica et al. (2013) described crystal family .Is a family for teams of different sizes and types (clear, yellow, orange, red, and blue) The popularone is crystal clear, which focuses on communications between team members and suppliers. Also, the team is small with seven characteristic objectives: frequent delivery, reflexive improvement, osmotic communication, personal safety, easy-to-access expert users according to their user story and story point, and concentration.

While Kumar and Bhatia (2014) and Tumbas and Matković(2006) and Aitken and Ilango (2013) disscussed the main goal of using FDD by decomposing the iteration has two stages: design and development. Trimble and Webster (2013) and Aitken and Ilango (2013) and introduced LSD as away of reduce cost and eliminate waste . Stoica et al. (2013) and Aitken and Ilango (2013) reviewed the DSDM and they focus on goal before the project starts, early testing, and high level of communication. İn addation Aitken and Ilango (2013) presented that customer involvement in traditional methodologies is Not involved and in AM is involved. Stoica et al. (2013) reviewed that Architecture in traditional methodologies is Design for the current and next requirement and in AM Design for the current requirements.Project domain decribed by Stoica et al. (2013) and Tumbas and Matković (2006) the organizations can use traditional methodologies in Large-scale system and AM in Small and medium projects. team members described by Aitken and Ilango (2013) and Tumbas and Matković (2006) traditional methodologies team members Mostly experts on models (how, which,where)to be used), AM team members as Cross-functional team members having experience with such problems by using the documentations (problem story, product story, user story).

Communication presented by Aitken and Ilango (2013) and Tumbas and Matković (2006) and Trimble and Webster (2013) in traditional methodologies Formal and in AM Informal (short daily meetings). Also customer value discussed by Stoica (2013) and Yadav (2015) and Trimble and Webster (2013) and Soundararajan and Arthur (2009) and Käpyaho and Kauppinen (2015) in AM is high and increase customer value and in traditional methodologies in agood

manner.Testing discussed by Aitken and Ilango (2013) and Käpyaho and Kauppinen (2015) in traditional methodologies for example ;prototyping or other modeling to make sense of requirements, test driven development (TDD) to ensure quality and review meetings and acceptance testing.AM In each iteration all the requirements will be tested to give feedback documentations.

Käpyaho and Kauppinen (2015) and Aitken and Ilango (2013) and Soundararajan and Arthur (2009) descibed validation and veryfication in traditional methodologies P can help to mitigate future risk by validating requirements before implementation;AM no detailed requirements documented to do a formal verification against writing tests first requires low-level refinement of specifications beforehand.Also Aitken and Ilango (2013) and Tumbas and Matković (2006) and Trimble and Webster (2013) reviewed that SQA in traditional methodologies documented requirements, without all details. Requirements are identified at thebeginning of devlopment and they change only exceptionally; in AM Improve software quality and responsiveness to changing customer requirements through simplest coding solutions.

Stoica (2013) and Tumbas and Matković (2006) disscussed theteam knowledge in traditional methodologies Expert team AM,Professional team with expertise about problem domain models and implementations.Aitken and Ilango (2013) and Yadav (2015) also Aitken and Ilango (2013) and Tumbas andMatković (2006) and Käpyaho and Kauppinen (2015) discussed Documentation in traditional methodologies Document driven. Every activity is measured by intensive documentation ordeliverables Code and AM Documentation is important in AM to reduce time and cost, such as product backlog, user story, product story and so on .

Aitken and Ilango (2013) and Käpyaho and Kauppinen (2015) reviewed Encourages in traditional methodologies Upfront architecture (with justification and evaluation of options). Formation of many models for specification.Rmergent and evolving architecture, specific guidelines, continuous refactoring and self_motivation.Adaptive to change has been reviewed by Aitken and Ilango (2013) ; Tumbasand Matković, (2006) and Trimble and Webster (2013). Activates work products are reviewed by Tumbas and Matković (2006) to enhance the processes of SDLC.The requirement for the combination framework as showed by the authors is fundamental; software development must guarantee they actualize the most suitable strategy that makes their items and administrations competitive. Obviously, the

advancement of the combination framework as traditional methodologies and AM during the SDLC has been significantly centered around making the SDLC successful, productive and less expensive.

CHAPTER 5

THE BENEFIT OF USING BOTH AM AND TRADITIONAL METHODOLOGIES AS A COMBİNATİON FRAMEWORK

This chapter will conclude the result and the main objective of using the combination framework and the future work .

5.1. Conclusıon

Software development these days presents the utilization of traditional software or agile methodologies (AM) to improve software development life cycle (SDLC). Some software development associations and organizations utilize both as a framework (traditional methodologies and agile methodologies), To pick up the optimal solutions. In addition, the purpose of the framework strategy is to make time to market and deliver fast within budget. The blend of the mix framework amongst traditional methodologies and AM environment that uses the upsides of both. So both as a philosophy attempts to enhance and upgrade SDLC and increment the opportunity to market. The fundamental point amid utilize the blend framework is to incorporate the system in the organization structure or the organization to be appropriate for the mix framework. Also, organizations begin to utilize such framework to abuse the potential helpful uses of the properties of both AM and traditional methodologies. So AM team members is self organization with serious correspondence in the framework they can change their organization structure keeping in mind the end goal to adjust to change; by utilizing this practices, the unit errands will be done quickly than regular.

In spite of the fact that there are a few difficulties that, there is no useful of joining the most recent century of utilizing the mixes framework, some organization approved that the combination framework is a method for making the commercial center additionally the customer value can have expanded significantly. A perfect way to deal with upgrade programming improvement gainfulness and quality are to focus on people. Agile software development ASD Framework can be used in a small project environment item change the methodology is people driven, correspondence organized, and willing to acclimate to expected or amazing changes

at whatever point. It progresses the incremental and iterative development of the item in little releases, and AMs can build costs and improving the nature of software items. The AMs Framework for Small Projects addresses specifics with respect to the usage of AMs and the appraisal of the results. In this review, a large portion of the papers are completed and sent proficiently in light of the mixed framework or correlation between the AMs and traditional methodologies techniques. This review exhibited that the structure demonstrated an abnormal state of efficiency enhance SDLC.

The framework is arranged due to the relationship and examination of a large portion of the agile methodologies including Scrum, XP, FDD, DSDM, and Crystal Clear.in expansion, practices and the CSFs in deft software wanders. The mixed framework then again obliges the "gigantic arrangement ahead of time" character from ordinary procedures. The offers general lithe practices that can be used for a wander and also a large portion of the traditional methodologies can be utilized as well. I solidly trust that it is important to test the defined speculation by pragmatic confirmation from comparable strategies or genuine utilization of the mix structure.

5.2.Future work

In this research present the main aim of using traditional methodologies and AM and how to use both as an optimal solution .this study aimed to introduce the major objective of the combination framework, and to present how can many different organization gain the advantages of using such kind of this framework.So in the future , the work of this kind of study will be more effective if the collected data is from practical side and taken real information from difrent kinds of organizations ,So how they are using this kind of framework or interest is one of this methods in their projects. Beside to this study is the practices of each method and which kind of each practices can be helpful during the combination of both methodologies.

Finally,I strongly belive that if the comprason study is made about real software industries that are using the combination framework or both to enhance SDLC.

REFERENCES

Abrahamsson, P., Babar, M. A., & Kruchten, P. (2010). Agility and architecture: Can they coexist?. *IEEE Software*, *27*(2).

Aitken, A. (2014). Dual Application Model for Agile Software Engineering. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 4789-4798). IEEE.

Aitken, A., & Ilango, V. (2013). A comparative analysis of traditional software engineering and agile software development. In System Sciences (HICSS), 2013 46th Hawaii International Conference on (pp. 4751-4760). IEEE.

Aoyama, M. (1996). Beyond software factories: concurrent-development process and an evolution of software process technology in Japan. *Information and Software Technology*, *38*(3), 133-143.

Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*.

Bannink, S. (2014). Challenges in the Transition from Waterfall to Scrum–a Case study at Port Base. In 20th Twente Student Conference on Information Technology.

Boehm, B., & Turner, R. (2003). *Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents*. Addison-Wesley Professional.

Boehm, B. W. (1981). *Software engineering economics* (Vol. 197). Englewood Cliffs (NJ): Prentice-hall.

Bishop, D., & Deokar, A. (2014). Toward an Understanding of Preference for Agile Software Development Methods from a Personality Theory Perspective. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 4749-4758). IEEE.

Delworth, T. L., Rosati, A., Anderson, W., Adcroft, A. J., Balaji, V., Benson, R., ... & Vecchi, G. A. (2012). Simulated climate and climate change in the GFDL CM2. 5 high-resolution coupled climate model. Journal of Climate, 25(8), 2755-2781.

Dreesen, T., Linden, R., Meures, C., Schmidt, N., & Rosenkranz, C. (2016). Beyond the Border: A Comparative Literature Review on Communication Practices for Agile Global Outsourced Software Development Projects. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on* (pp. 4932-4941). IEEE.

Ebert, C., Abrahamsson, P., & Oza, N. (2012). Lean Software Development. IEEE Software, 29(5), 22-25.

Feng, Y., Liu, Y., Wang, X., Dong, D., Shi, Y., Hua, S., ... & Tang, L. (2016). Compact Nanofilters Based on Plasmonics Waveguide With Archimedes' Spiral Nanostructure. *IEEE Photonics Journal*, *8*(5), 1-8.

Glaiel, F. (2012). Agile project dynamics: A Strategic project management approach to the study of large-scale software development using system dynamics (Doctoral dissertation, Massachusetts Institute of Technology).

Hui, A. (2013). Lean change: Enabling agile transformation through lean startup, kotter and kanban: An experience report. In *Agile Conference (AGILE), 2013* (pp. 169-174). IEEE.

Huang, X., Wang, X., & Liu, C. (2001). Evaluation of agile manufacturing enterprises. Tsinghua Science and Technology, 6(1), 38-41.

Highsmith, J. A. (2002). Agile software development ecosystems (Vol. 13). Addison-Wesley Professional.

Jiang, L., & Eberlein, A. (2008). Towards a framework for understanding the relationships between classical software engineering and agile methodologies. In *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral* (pp. 9-14). ACM.

Käpyaho, M., & Kauppinen, M. (2015). Agile requirements engineering with prototyping: A case study. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International* (pp. 334-343). IEEE.

Kumar, G., & Bhatia, P. K. (2014). Comparative analysis of software engineering models from traditional to modern methodologies. In *Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on* (pp. 189-196). IEEE.

Kuusinen, K., Gregory, P., Sharp, H., & Barroca, L. (2016). Strategies for doing Agile in a non-Agile Environment. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (p. 5). ACM.

Kakar, A. K. (2014). Teaching Theories Underlying Agile Methods in a Systems Development Course. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 4970-4978). IEEE.

Liu, C., Yang, Z., Wang, X., & Huang, X. (2003). Manufacturing partner selection by network based on "Invite/submit tenders". *Tsinghua Science and Technology*, *8*(1), 97-104.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72-78.

Palmquist, M. S., Lapham, M. A., Miller, S., Chick, T., & Ozkaya, I. (2013). *Parallel worlds: Agile and waterfall differences and similarities* (No. CMU/SEI-2013-TN-021). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Richter, I., Raith, F., & Weber, M. (2016). Problems in Agile Global Software Engineering Projects especially within Traditionally Organised Corporations:[An exploratory semi-structured interview study]. In *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering* (pp. 33-43). ACM.

Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012, September). Survey on agile and lean usage in finnish software industry. In *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on* (pp. 139-148). IEEE.

Schmalz, M., Finn, A., & Taylor, H. (2014). Risk Management in Video Game Development Projects. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 4325-4334). IEEE.

Soundararajan, S., & Arthur, J. D. (2009). A soft-structured agile framework for larger scale systems development. In *Engineering of computer based systems, 2009.*

ecbs 2009. 16th annual ieee international conference and workshop on the (pp. 187-195). IEEE.

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4), 64.

Sumrell, M. (2007). From waterfall to agile-how does a QA team transition. In Proceedings of AGILE (pp. 291-295).

Tumbas, P., & Matkovic, P. (2006). Agile vs. Traditional methodologies in developing Information Systems. Management Information Systems, 1(2006), 15-24.

Pikkarainen, M., Conboy, K., Karlstöm, D., Still, J., & Kerievsky, J. (2009). What Skills Do We Really Need in Agile Software Development?–Discussion of Industrial Impacts and Challenges. In International Conference on Agile Processes and Extreme Programming in Software Engineering (pp. 267-270). Springer Berlin Heidelberg.Trimble, J., & Webster, C. (2013, January). From traditional, to lean, to agile development: Finding the optimal software engineering cycle. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on* (pp. 4826-4833). IEEE.

Yadav, M., Goyal, N., & Yadav, J. (2015, March). Agile Methodology over iterative approach of Software Development-A review. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on* (pp. 542-547). IEEE.

APPENDIX -A

Table A-1. The table 1 is describing The used framework and evaluations

Authors name	Actual framework as optimal solution	Type AM	Traditional methodologies	Benefits	Disadvantages
(Käpyaho and Kauppinen, 2015)	Combination of AM with Prototyping (building Acceptance Test Driven development)	ASD	Prototyping	1) Helps in solving some of the agile requirement engineering challenges 2) Solves some issues often encountered in agile projects 3) Prototyping is most beneficial for agile projects with some complementary practices 4) Act as tangible plans to accept adaptation 5) Builds motivation 6) Builds Acceptance Test Driven development	1) Difficulty improving optimal communication between team members and stakeholders 2) Prototyping cannot solve certain problems in ASD 3) Partially optimized and volatile requirements in agile practices 3) Prototyping leads to worsening of the problems in architectural requirements and neglecting quality 4) Lack of broader pictures and quality requirements
(Bannink, 2014)	Water- scrum -Fall	scrum	Waterfall	1) Faster time-to-market 2) Increased business value 3) Improved flexibility 4) Responsiveness 5) Includes project planning and budget 6) Early feedback 7) Leads to team autonomy 8) Effective communication	1) The transition to scrum needs full support of upper management (related to sponsorship). 2) Impediments originating from in terms of feedback 3) Lack of commitment 4) Requires a shift in the mindset

				9) Transition to scrum is an improvement 10) Design upfront takes less time 11) High-quality product 12) Multi-skilled teams can emerge 13) User story and story point lead to accurate estimations	
(Trimble and Webster, 2013)	Moving from traditional methodologies to LM to AM	LM and scrum	traditional methodologies (Prototyping and Waterfall)	1) Increase customer and developer satisfaction 2) The team will have a higher priority 3) Designing and building the most important features of the product 4) Eliminating waste 5) Accept changes during short delivery cycles	Ranked list functions
(Soundararajan and Arthur, 2009)	Framework (soft-structured requirement and a tailored development) Requirements Generation Model	ASD	Waterfall	1) Accepts changes 2) Increases customer value 3) Short iterations lifecycles 4) Embodies philosophy to the traditional requirement engineering processes 5)Increases agility value 6) Determines the end of each iteration by well-defined span activities	
(Sumrell, 2007)	Traditional automated testing strategy to scrum framework	scrum	traditional methodologies	1) Uses expert outsources 2) Defines team roles 3) Starts to use Fitness (light-weight automated	1) Shifting from waterfall to agile is difficult, sometimes leading to failure 2) Long cycles of regression testing

				testing tools) 4) Understands and recognizes the skills of Team Members in test automation 5) Eliminates re-work	3) Non- experimental team practices regarding testing software
(Kumar and Bhatia, 2014)	Comparative analysis of traditional methodologies and AM	XP, scrum, FDD	Waterfall, spiral model, iteration, Rapid Application Development, V-Model, RUP	1) Light-weight documentation 2) Cost reduction 3) Maintainable Software Product 4) Rapid AM, iterative, evolutionary, adaptable to change; Waterfall is a reusable component; can be used to reduce development; spiral model easy to understand and implement; Rapid Application Development can consider rapid feedback 5) Cost and time reduction with both traditional methodologies and AM 6) Flexible scrum 7) Increased performance by AM	1) Traditional methodologies is not efficient in managing rapid change 2) Traditional methodologies rather predictive than adoptive to change 3) Traditional methodologies has hard time to deal with change 4) Waterfall is not applicable in large projects. 5) Waterfall can hardly integrate risk management 6) Waterfall is not adoptive to rapid change 7) V-Model cannot provide a clear path for problems during the testing phase 8) spiral model does not work well with small projects; models are costly to use 9) Traditional methodologies needs a sizeable number of developers
(Richter, 2016)	Applying AM in organized corporations; traditional	AM	traditional methodologies organized	Good and constant communication leads efficiency and support from the top management	1) The team can mistake the use of AM in the project 2) Difficult expectations in different

	methodologies semi-structured				AM projects 3) Problems related to human factor 4) Lack of expectation on the customer's side due to lack of support
(Kuusinen et al., 2016)	Combination framework	AM	traditional methodologies	1)The organization can apply bottom-up change 2) AM leads to implementing decomposition tasks 3) Tailors agile to suit a hierarchical organization 4)Increases productivity and team performance	1) Company reward systems are not suitable for AM and IT system 2) Novice team may not know how to achieve change from within the organization 3) Hand-over projects 4) Project team might not precisely follow AM 5) The management may not understand agile 6) The team might not self-manage
(Yadav, 2015)	Comparative AM and traditional methodologies	AM	Iteration, RUP	1) AM increases the chances to make the project successful; monitors all the processes during each iteration to allow the customer to know everything about the Software Product; AM includes Iteration and incremental; can fix bugs quickly; 2) Iteration is small waterfall, lack of time because of impossible fixing bugs. 3) Rapid P builds dirty code; insufficient test; improves delivery time; increases susability and productivity	1) Adoption is difficult for ITER and Waterfall 2) Lack of documentation with the customer and development team 3) diminished control by management

				4) AM satisfies additional needs over Iteration	
(Aitken, 2014)	framework to support Dual application model (the best of AM with the best of traditional methodologies)	AM	traditional methodologies	1) Meets both functional non and -functional requirements according to their domain 2)ASD code covers most of the physical aspects and solutions 3) Logical model is the solution for the problem domain and software solution 4) Dual application focuses on defining the problem as related to product owner, domain experts and implementation	1) In traditional methodologies, multitude of models can confuse the team and need decision as to which one is to be chosen 2) Agile developers can have informal arguments 3) Developers might change the code without updating the model
(Stoica, 2013)	AM and traditional methodologies framework	AM	Traditional methodologies	1) More success in the market 2) Delivery in time and within budget 3) Accept continuous adaptation with less cost 4) Increased customer value 5) Waterfall documentation and structure is a plus when new members join the team 6) Incremental model easy to test and debug in small iterations; flexible with less cost 7) Increased organizational ability to manage the market and optimize use of HUMRES 8) scrum increases productivity	1) HUMRES 2) Non-expert team cannot know how, when and where to use SDLC models
(Aitken and Ilango, 2013)	Comparative AM and traditional methodologies	AM	Traditional methodologies	1) Improves all project outcomes 2) Improves productivity, efficiency and effectiveness 3)Empowers teams to take responsibility for	

				planning iterations (or releases) and delivering on those plans.	
(Tumbas and Matković, 2006)	Comparative traditional methodologies andAM	XP; scrum; FDD; DSDM; Adaptive Software Development; LSD; crystal family	Traditional methodologies	1) AM define user requirement 2) AM deliver money value 3) Traditional methodologies deal with future solution 4) AM give verbal and frequent communication	

Table A-2. The described framework combination or compression framework.

Authors	Combination framework
• (Käpyaho and Kauppinen, 2015)	• AM+ Prototyping
• (Bannink, 2014)	• Waterfall + scrum
• (Trimble and Webster, 2013)	• Traditional methodologies +LM+AM
• (Soundararajan and Arthur, 2009)	• AM requirements + Traditional methodologies requirement
• (Sumrell, 2007)	• scrum + Waterfall
• (Kumar and Bhatia, 2014)	• XP, scrum, FDD - Waterfall, spiral model, Iteration, Rapid Application Development, V-Model, RUP
• (Richter, 2016)	• AM practices + traditional methodologies
• (Kuusinen et al., 2016)	• AM+ traditional methodologies
• (Yadav, 2015)	• iteriative, AM, RUP
• (Aitken, 2014)	• AM+ traditional methodologies
• (Stoica, 2013)	• AM+ traditional methodologies
• (Aitken and Ilango, 2013)	• AM, traditional methodologies
• (Tumbas and Matković, 2006)	• XP; scrum; FDD; DSDM; Adaptive Software Development; LSD; crystal family, traditional methodologies

Table A-3. The combination framework that can be used in project domains.

Authors	The combination framework	The project domains	The practices
(Käpyahoand Kauppinen, 2015)	Combination agile with Prototyping (building Acceptance Test Driven development)	LPs, Complex Software Project Development, Agile Projects Short-term projects , easy-to-use UI for structural applications and small projects Structured applications in short life cycle projects	• Communication • Documentations and light wight methodology • continuous refactoring • continuous activity • AM require JIT
(Bannink, 2014)	Water- scrum -Fall	1) The development process (previous content here was rather a disadvantage, hence deleted Also check the rest) 2) Larger software projects (healthcare, emergency) 3) Applications that high severity in healthcare 4) Sequential waterfall model for larger software projects to control over the output 5) Software engineering and project management 6) Waterfall for projects that are not subject to changes in requirements 7) Scrum for projects that use the scrum Master.	• Quantifying agile practices • Communication

		Also to define the Product Owner which linked to a project backlog.	
(Trimble and Webster, 2013)	Moving from traditional methodologies to LM to AM	1) User centric software platform 2) Projects require short delivery life cycle (short term projects) 3) Small projects 4) Multiple projects 5) In NASA software projects	• Early delivery • continuous integration • continuous feedback
(Soundararajan and Arthur, 2009)	soft-structured requirement and tailored development	1) Large-scale complex systems 2) E-commerce company 3) Small scale systems 4) Large and small projects 5) Financial projects	• Less documentation • Refactoring (not suitable for large- scale system development) codes • Upfront planning • Just-in-time • validation and veryfication • TDD (not suitable for large-scale system development) • AM practices
(Sumrell, 2007)	Traditional automated testing strategy to scrum	1) Applications used for SQA large and meduiem projects	• Early delivery • Continuous testing

	framework	2) Emergency projects and Misys Healthcare Systems 3) Projects require high severity equipments 4) Intensive care room 5) projects require the functional, system and regression testing 6) In projects that used highly skilled group of automation testers using traditional methodologies, expensive off the shelf automated testing software	• SQA
(Kumar and Bhatia, 2014)	Comparative analysis of traditional methodologies and AM	AM used to improve projects by ensuring short time cycle SQA; spiral model used in government projects in many companies; Rapid Application Development for small projects; spiral model can be used in large and mission-critical projects	Decision support system (the most important team (most important team select SDM that best suit the project)
(Richter, 2016)	Apply AM in organized corporation traditional methodologies (semi-structured)	Structured applications and projects requiring rapid change Short projects software development projects displaying highly severity failures by using XP	In AM pair-programming when using XP
(Kuusinen et al., 2016)	Combination framework	scrum of scrums for large distribution projects and R- scrum for regulated environments and specific	AM and traditional methodologies practices

		organizational contexts; in large organizations and projects with rapid change Large projects	
(Yadav, 2015)	Comparative (AM, traditional methodologies)	Industrial organizational projects requiring business and IT Software for computerized systems Software for managing the distribution of advertising pamphlets	AM practices and continuous delivery,
(Aitken, 2014)	framework to support Dual Application model (the best of AM with the best of traditional methodologies)	1) Larger and more real-world software development projects 2) a single-user desktop application 3) Web-based application for multiple users 4) Software for internet applications	traditional methodologies practices
(Stoica, 2013)	AM and traditional methodologies framework	1) Complex business environments 2) Architectural applications 3) Waterfall for small projects. 4) In projects relying upon business process management	Continuous improvement and team self-organisation Self- Organization team, continuous integration

		5) XP can be used in structural applications and to evoid failure for software project that used in high degree of severity 6) Economic projects 7) Basic plan projects 8) Traditional methodologies for large and medium projects 9) In projects that built by motivated and credible persons 10) Software projects that built by usability 11) Projects with dynamic requirements 12) AM for small and medium projects 13) traditional methodologies for projects at hand 14) Waterfall complex and object oriented projects.	
(Aitken and Ilango, 2013)	Comparative AM and traditional methodologies	1) Architectural applications 2) Projects that produce high-quality products	/////
(Tumbas and Matković, 2006)	Comparative AM and traditional methodologies	1) project information systems 2) AM large scale system 3) developing information system for software	AM practices, Activates work products

		projects 4) management information systems 5) XP in complex projects 6) projects are including many changes (adoptive to change) 7) FDD for large projects with long life cycle 8) In small projects where customer and developers are close to each other (good communications) In internal projects In company or organization that their team members not experienced enough DSDM in projects applications that satisfies user's requirements by determining the functionality	

Table A- 4. The main differences between traditional methodologies and AM .

Attributes	traditional methodologies	AM	Authors name
Architecture	Design for the current and next requirement	Design for the current requirements	▪ (Stoica, 2013)
Project domain	Large-scale system	Small and medium projects	▪ (Stoica, 2013) ▪ (Tumbas and Matković, 2006)
Team size	Large organized team	Small team (5-10)	▪ (Stoica, 2013) ▪ (Tumbas and Matkovi, 2006)
Team Members	Mostly experts on models (how, which, where) to be used)	Cross-functional team members having experience with such problems by using the documentations (problem story, product story, user story)	▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković, 2006)
Communication	Formal	Informal (short daily meetings)	▪ (Aitken and Ilango, 2013) ▪ (Tumbas andMatković, 2006) ▪ (Trimble and Webster, 2013)
Remodeling	Expensive	Low cost (adaptive to change)	▪ (Tumbas and Matković, 2006)
customer value	In a good manner	High manner	▪ (Stoica, 2013) ▪ (Yadav, 2015) ▪ (Trimble and Webster, 2013)

			▪ (Soundararajan and Arthur, 2009) ▪ (Käpyaho and Kauppinen, 2015)
Customer involvement	Low	High (throughout monitoring the project processes)	▪ (Stoica, 2013) ▪ (Trimble and Webster, 2013) ▪ (Soundararajan and Arthur, 2009) ▪ (Käpyaho and Kauppinen, 2015)
Testing	prototyping or other modeling to make sense of requirements, test driven development (TDD) to ensure quality and review meetings and acceptance testing	In each iteration all the requirements will be tested to give feedback documentations	▪ (Aitken and Ilango, 2013) ▪ (Käpyaho and Kauppinen, 2015)
validation and veryfication	Prototyping can help to mitigate future risk by validating requirements before implementation	No detailed requirements documented to do a formal verification against Writing tests first requires low-level refinement of specifications beforehand	▪ (Soundararajan and Arthur, 2009) ▪ (Aitken and Ilango, 2013) ▪ (Käpyaho and Kauppinen,2015)
Suppliers	The developers explore possible solutions and use an iterative/incremental approach to build a prototype and send feedback.	The suppliers are self-management and self-organization so they can change the structure of work instead adoptive to change	• (Trimble and Webster, 2013)
User requirements	user can give his comments not until the end of the developmental	will defined	▪ (Tumbas and Matković, 2006)

	cycle		▪ (Käpyaho and Kauppinen, 2015)
SQA	documented requirements, without all details. Requirements are identified at the beginning of devlopment and they change only exceptionally.	Improve software quality and responsiveness to changing customer requirements through simplest coding solutions.	▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković, 2006) ▪ (Trimble and Webster, 2013)
Organization structure		Teams are self_organized, with intensive communications in the framework	
Cost	High	low	▪ (Tumbas and Matković, 2006)
Project level		Large scale projects	▪ (Tumbas and Matković, 2006)
Re-working	High	low	▪ (Tumbas and Matković, 2006)
Management trend	Command and control	Self-management	▪ (Tumbas and Matković, 2006)
Leadership	Command and control	Good leadership	▪ (Käpyaho and Kauppinen, 2015)
Planning	Before the start, there is plan documentation.	In each iteration, there is a plan to get feedback and know how to start the next iteration	▪ (Stoica, 2013)
Team knowledge	Expert team	Professional team with expertise about problem domain models and implementations	▪ (Stoica, 2013) ▪ (Tumbas and Matković, 2006)

Internal resources	The team must be advanced and expert on traditional methodologies models	Used for decisions with cross-functional teams of high quality and use some resources	▪ (Tumbas and Matković, 2006)
External resources	Used when there is a need for expert knowledge (sub-contractor).	Used when there is a need for expert knowledge (sub-contractor). Also recommended that he should be from outside of the project	▪ (Tumbas and Matković, 2006)
Contract review	Must be crystal clear between the suppliers and the customer. estimating lines of code and making creating detailed estimates for to what extent it would take to build up every key list of capabilities, then utilizing those assessments as a contract with client.	The contract review after negotiation can be helpful and eliminate risk in the future and include two items (the third will be discussed next) into the contract.	▪ (Aitken and Ilango, 2013) ▪ (Trimble and Webster, 2013)
Documentation	Document driven. Every activity is measured by intensive documentation or deliverables. Code	Documentation is important in AM to reduce time and cost, such as product backlog, user story, product story…etc..	▪ (Aitken and Ilango, 2013) ▪ (Yadav, 2015) ▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković,2006) ▪ (Käpyaho and Kauppinen, 2015)
Iteration	short iteration lengths, user stories were divided into tasks	short iteration lengths	▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković,2006) ▪ (Käpyaho and Kauppinen, 2015)
Encourages	Upfront architecture (with justification and evaluation of options). Formation of many models for specification	emergent and evolving architecture, specific guidelines, continuous refactoring	▪ (Aitken and Ilango, 2013) ▪ (Käpyaho and Kauppinen, 2015)

Constant customer involvement	Not involved	Involved	▪ (Aitken and Ilango, 2013)
Adaptive to change	Does not welcome change	Welcomes rapid change	▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković,2006) ▪ (Trimble and Webster, 2013)

Table A-5. Types of AM and the attributes .

AM	Attributes	Authors
XP	A programming-focused software development method expand software quality and responsiveness to modified customer requirements by means of simplest coding solutions. An easy methodology for small to medium teams that develop software products with vague or changing requirements**.** Inspired by RUP, XP has six stages and addresses the value of community, simplicity, feedback and courage. XP involves some practices such as planning, TDD, customer involvement, refactoring, pair programming, and continuous integration.	▪ (Stoica, 2013) ▪ (Kumar and Bhatia, 2014) ▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković, 2006)
scrum	Involves self-management and self-motivation in teams with effective communication between the scrum master and the stakeholder to receive useful feedback upon short daily meetings that last 15 minute. in each iteration. Also, the features that must be implemented are recorded in a list of unsolved order and the client decides which orders will be implemented in the next sprint. The scrum master is the person who is responsible for solving any active problems and empowering the team members during the sprint.is focusing on project management, define the product backlog activities	▪ (Stoica,2013) ▪ (Sumrell, 2007) ▪ (Bannink, 2014) ▪ (Kumar and Bhatia, 2014) ▪ (Trimble and Webster, 2013) ▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković,2006)
crystal family	Is a family for teams of different sizes and types (clear, yellow, orange, red, and blue) The poplar one is crystal clear, which focuses on communications between team members and suppliers? Also, the team is small with seven characteristic objectives: frequent delivery, reflexive improvement, osmotic communication, personal safety, easy-to-access expert users according to their user story and story point, and concentration,	▪ (Stoica, 2013) ▪ (Aitken and Ilango,2013) ▪ (Tumbas and Matković,2006)
FDD	Is suitable for most of the projects. emphasizing initial object model, work division into features and the iterative design of each feature. The iteration has two stages: design and development.	▪ (Kumar and Bhatia, 2014) ▪ (Aitken andIlango, 2013) ▪ (Tumbas and Matković,2006)
LSD	Lean Software Development focuses on reducing cost, delivering early and within budget, and eliminating waste. Lean has seven principles: (to eliminate waste, deliver fast, delay commitment, amplify learning,	▪ (Trimble and Webster, 2013) ▪ (Aitken and Ilango, 2013)

	empower the team, build integrity, and see the whole.	▪ (Tumbas and Matković,2006)
DSDM	This method divides the project into 3 stages: pre-project, project life cycle, and post-project and contains nine principles: user involvement, team empowerment, early delivery, handling current needs, Iteration and incremental development, change review allotment, focus on goal before the project starts, early testing, and high level of communication.	▪ (Stoica, 2013) ▪ (Aitken and Ilango, 2013) ▪ (Tumbas and Matković,2006)