Quality Function Deployment Matrix Based for Agile Large-Scale Software Development

استخدام مصفوفة وظائف الجودة (QFD) في تطوير البرمجيات السريعة واسعة النطاق

Student: Amro “Mohammad Hani” “Al-Said Ahmad”

Supervisor: Dr. Akram Othman AlMashaykhi

A Thesis Submitted in Partial Fulfillment of Requirement for the Degree of Master Science in Computer Science

Amman Arab University

Faculty of Computer Science and Information

2014

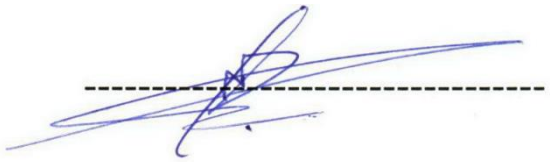
Authorization Statement

I, Amro M. Hani Al-Said Ahmad, authorize Amman Arab University to Supply hard and electronic copies of my thesis to libraries, establishments, or institutions concerned with research and scientific studies upon request, according to the university regulations.

Name: Amro M. Hani Al-Said Ahmad

Date: 05/02/2014

Signature: ----------



Resolution of the examining committee

This thesis titled "Quality Function Deployment Matrix Based for Agile Large-Scale Software Development" .Has been defended 22/1/2014.

and approved on 1/2/2014.

Examining Committee	Title	Signature
Prof. Dr.Alaa Al-Hamami	Chair	Alhamami
Dr. Akram Othman	Member and Supervisor	A.M Othman
Prof. Dr. Said Al-Goul	Member	Algoul

Acknowledgments

I would like to express my sincere gratitude to the following for their assistance during the course of this study. **Dr. Akram Othman**, my supervisor for his great support, encouragement, and for his academic views comments during the entire study. I am really thankful. And also special thanks to **Dr Said Al-Ghoul** and **Ala'a Al-Hamami** for make this work better.

I would like also to express my deep gratitude to those who were the reason behind this work and all what I have managed to achieve so far, **My Family (father, mother, brothers, and my sweet sister)**. And I would like to say thank you for their help to achieve my goal through their patience, love, understanding and willingness to accommodate my dream.

Special thanks also due to **Dr. Isam Najib**, for his valuable assistance, and all advices that he give to me, which was of significant effect on the successful completion of this thesis. I am deeply grateful to him for his encouragement and support. Also special thanks to all My Friends and Colleagues.

To all the above, I would like to say Thank you for that, without your assistance and support, the production of this work could not be possible.

Abstract

Quality Function Deployment Matrix Based for Agile Large-Scale Software Development

By: Amro Mohammad Hani Al-Said Ahmad

Supervisor: Dr. Akram Othman

Many challenges and obstacles have been arising when Agile methods are being used in large-scale projects. Besides that the quality is one of the key factors to measure the success of the project, this research introduce doesn't methods to solve those challenges only, but to ensure that the development organization has all what it needs to complete the project under using Agile methods in large-scale project and introduce methods to improve the quality and the relationship between all stockholders.

The scope of this study is to gain understanding how to facilitate the process to adopts Agile large-scale projects, that gives a better look and understood the best ways to avoid challenges and obstacles. Also it gives recommended processes to create an infrastructure to help the project manager to facilitate scaling Agile in large projects.

Those recommended processes attempt to help the project managers in a number of ways to support them, to enhance, and to improve the communication and the coordination between the team members, to select the right infrastructure for their projects, to know how to deal with risks and quality assurance in large-scale projects, and to know how to facilitate large-scale development, and how to deal with change management in Agile large-scale projects.

On the other hand, this study offers a solution for the process of dealing with Agile large-scale projects challenges. It presents a way that highlights on using modern communications technologies and social media websites in the process of maintaining the element of permanent connection between the team members, and it helps sharing the domain knowledge between team's member to deal with the social skills, Agile training, documentation, decision making, and testing.

This study uses the Quality Function Deployment (QFD) matrix tool as a method to improve the quality and help the project manager in large-scale projects. QFD is a method of customer oriented product developments and it has been used in a various kind of industries. This makes it one of the most important methods to improve the relationship with the customer.

Key words: Agile large-scale Development, QFD, Process Improvement, Infrastructure for Agile large-scale.

ملخص

"استخدام مصفوفة وظائف الجودة (QFD) في تطوير البرمجيات السريعة واسعة النطاق"

عمل: عمرو محمد هاني السيد أحمد

إشراف: الدكتور أكرم عثمان

ظهور العديد من التحديات و العقبات عند استخدام أحد منهجيات تطوير البرمجيات السريعة (Agile) في تطوير المشاريع ذات النطاق الواسع. وأيضا إن جودة المنتج التي تعتبر معيارا هاما وأساسيا لقياس نجاح المشروع، كانت أهم الأسباب لتقديم هذه الدراسة، فقدمت الدراسة ليس فقط حلولا لمواجهة التحديات التي تواجه البرمجيات السريعة و لكنها قدمت أيضا جميع الجوانب التي تحتاجها المؤسسة لإنهاء المشروع وتسليمه من خلال استخدام منهجيات البرمجيات السريعة في المشاريع ذات النطاق الواسع، وتقدم هذه الدراسة أيضا طرق تحسين نوعية الجودة و العلاقات بين جميع أطراف وأصحاب المصلحة في المشروع.

إن ما تشتمل عليه هذه الدراسة يحقق مكاسب عديدة على صعيد كيفية تسهيل و تبسيط عملية تبني منهجيات البرمجيات السريعة في المشاريع ذات النطاق الواسع، كما توفر لنا نظرة أفضل وفهما أوسع للطرق التي تساعدنا على تجنب التحديات و الصعوبات والتغلب عليها، كذلك تزودنا هذه الدراسة بمجموعة من العمليات الموصى بها والمحققة لخلق بنية تحتية لتطبيق استخدام البرمجيات السريعة في المشاريع ذات النطاق الواسع.

تلك العمليات الموصى بها هي محاولات لمساعدة مديري المشاريع في عدد من الطرق، لتقديم الدعم لهم لتعزيز وتحسين التواصل والتنسيق بين أعضاء الفريق، وكيفية اختيار البنية التحتية المناسبة لمشاريعهم، وكيفية التعامل مع المخاطر وضمان الجودة في المشاريع واسعة النطاق، وكيفية تسهيل التطوير في مشاريع النطاق الواسع، وايضا كيفية التعامل مع "إدارة التغيير" في المشاريع السريعة واسعة النطاق.

من ناحية أخرى، فإن هذه الدراسة تقدم حلا لعملية التعامل مع تحديات المشاريع واسعة النطاق باستخدام البرمجيات السريعة، ومحاولة تسلط الضوء على كيفية استخدام تكنولوجيا الاتصالات الحديثة ومواقع التواصل الاجتماعي في عملية الحفاظ على عنصر الاتصال الدائم بين أعضاء الفريق والزبائن، والمساعدة على تبادل معرفة المجالبين أعضاء الفريق، و تتناول الدراسة ايضا معالجة المهارات الاجتماعية والتدريب على البرمجيات السريعة والتوثيق، واتخاذ القرارات، والاختبار.

وقد تم استخدام مضمونة وظائف الجودة (QFD) كأداة لمساعدة تحسين وتطوير الجودة، ومساعدة مدراء مشاريع النطاق الواسع، حيث أنها أداة (مصفوفة) تقوم على التركيز على خدمة العملاء وتطوير المنتج وقد تم استخدامها في العديد من أنواع الصناعات والمشروعات الكبرى، وهذا يجعل من هذه الأداة من أهم أدوت تطوير المنتج وتحسين العلاقات بين الزبائن و القائمين على المشروع.

Table OF Contents

Contents

Table OF Figures

Table OF Tables

Chapter One
Introduction

Introduction

Software development is a complex process that can be achieved in different ways with different results. In a perfect way, to be a successful software developer, building a quality project in an identified time frame, budget and required functionality must be achieved. So there are many different engineering approaches for different situations. Some methods have been more popular in time; that known as Agile software development. Those methods are showing benefits to small and medium scale software systems and they shouldn't be ignored by global and large companies developing large-scale systems.

Since first emerged of Agile software development in the late 1990's have changed the landscape of software development; Agile software developments deal with the rapidly changing business environment in which most organizations operate is challenging traditional software engineering approaches. Software development organizations often must deal with requirements, business domain that tend to evolve quickly and become obsolete even the project not complete yet [Cao & Ramesh, 2008].

Agile thinking is a seeking to simplify things by decreasing complexity of planning, by focusing on customer value, and shaping a rich environment of sharing and collaboration. There are a large number of methods, techniques, best practices, that claim to be "Agile".

Agile methods seek to address the challenges and obstacles in such dynamic contexts that have gained much interest among practitioners and researchers. There for Agile methods blended technical and human behavioral together; that makes Agile methods as excellent way to deliver software faster and with higher quality when teams understand the unique challenges they face.

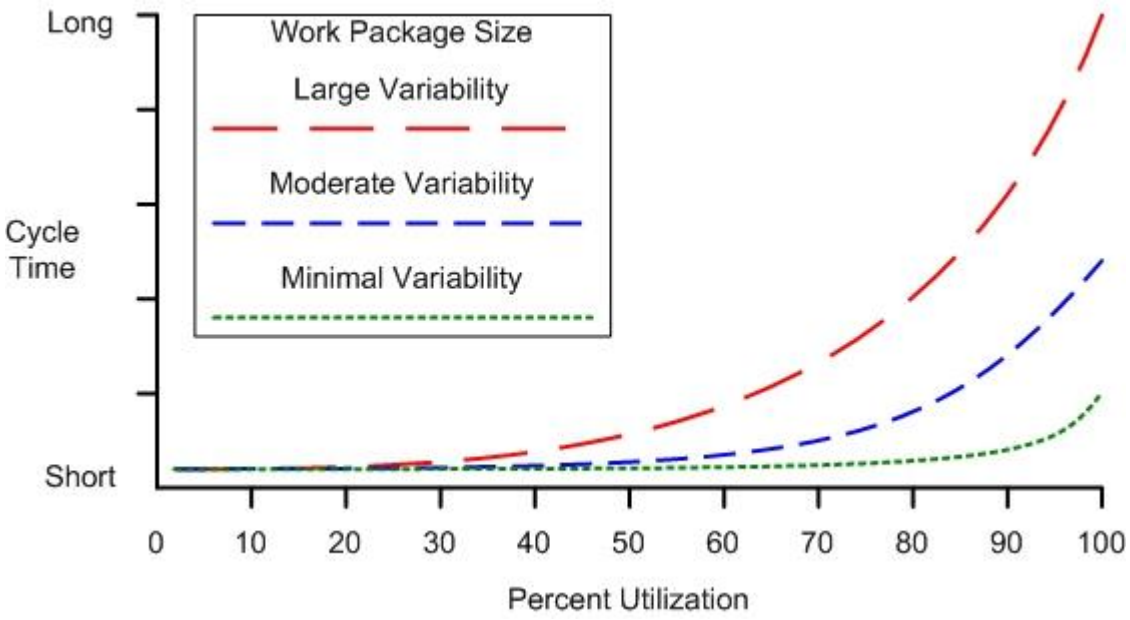


Figure1.1 shows that there is a regularity relationship between variability and the cycle time, so when variability increases, it will added more risk so it will increase the probability of occurring challenges.

Figure1.1: Effect on variation in package size on cycle time [Tenhunen, 2010, p.17]

1-1 Background:

This section reviews briefly two concepts, scaling Agile methods, and challenges and obstacle that have been faced.

Scaling Agile methods have two perspectives that been identified; the first one is a “scaling up” perspective; concerned using method for development large systems that cannot be develop by small team, the second perspective is a “scaling out”; concerned with how Agile methods can be introduced across large organization with many years of software development experience [Sommerville, 2011].

Through time the organizations start to scale Agile software development in small and medium way and they succeeded to overcome challenges and obstacles, and in that time none could predict how well it worked when applied at large scale.

When large organizations start to apply Agile for their software development projects, they start to understand more about the large-scale Agile software development, and that organizations have reported that there are challenges and obstacles in scaling Agile software development in large and global projects.

Ability to scale Agile software development to large organizations has always had skeptics. As Agile has become Mainstream many enterprises are initializing their own Agile adoption Agile Engineering approach that fit their needs [Tenhunen, 2010]. Nevertheless obstacles and challenges faced.

Therefore much challenges emerged in Scale software development. That caused number of reason such as; team coordination, knowledge sharing, facilitating self-management, communication, risk control, Team location, project testing, project management, decision making and Requirement Engineering.

We know that Agile scrum team is best used co-located team with five to nine team members. But with large development system the team starts to expand. There are some means of scaling Agile methods, rather than creating a single 70-80person team. The Agile Methods encourage splitting team member into multiple smaller teams. Planning a large, multiple team projects may require [Tenhunen, 2010]:

1- Establishing a basis how to estimate, teams that should have a common unit for estimating.
2- Adding details to user scenarios in early stages to understand the big problem target and to help the teams to coordinate their plan.
3- Executing look ahead planning so that teams can successfully work together.
4- Merging feeding buffers so that unexpected events in one team won't risk the whole project.

1-2 Quality Function Deployment (QFD):

QFD is a method for developing a design quality aiming at satisfying the stockholders and transforming their needs to measurable objectives of design quality targets and major quality assurance points to be used throughout the development phases. QFD is a way to assure the design quality while the product is still in the design stage [Akao, 1990].

QFD consists of two components: quality and function. The "quality deployment" component brings the stockholders needs into the design process. The "function deployment" component links different organizational functions and units into to the design-to-development transition via the formation of design teams [Lockamy & Khurana, 1995].

QFD has been evolved by product development people in response to the major problems in the traditional processes, which were [Clausing, 1994]:

- Ignoring the voice of customer
- Disregard the competition
- See each system specification is isolated
- Low expectations
- Little input from design and production people into product planning
- Divergent interpretation of the specifications
- Lack of structure
- Lost information during the processes
- Weak commitment to previous decisions

1-3 Problem Statement:

Since the 1990s, Agile software development methods have been effectively adopted at small and medium scale levels. At that time, none could predict how well it worked when applied at large-scale, through time Agile has become mainstream; many organizations have adopted their own relevant versions of Agile methods. Notwithstanding obstacles and challenges faced, this study tries to probe such challenges and obstacles emanating from people, development process, business environment, culture, policies, management, or a combination of all.

Following that stream of thinking leads to the overall goal of this research to go deeper, and tries to answer the following questions:

- What differences exist among small, and large scale Agile software development?
- What challenges and obstacles hinder the adoption of large scale Agile software development?
- What propagates obstacles affecting the use of large scale Agile methods of software development?
- How to overcome the obstacles facing large scale Agile software development?

The study attempts to adopt a new quality paradigm using “Quality Function Deployment Based for Agile large-scale Software Development” to transform the requirements and the needs of stakeholders using the Agile methods into measurable objectives of design quality. That assuredly helps in controlling the challenges and overcoming the obstacles faced upon adopting Agile methods at a large scale.

1-4 Reasons and Importance of this research

The researchers and the organizations consider Large-scale Agile software development is an active research area [Ambler, 2006]. Also there is a lack in studies that deal particularly with Agile in large-scale. There are certain studies attempt to adopt quality control matrix and put a recommendations to overcome the challenges and the obstacles that occur in large-scale Agile software development.

As known globally distributed software development and Agile methods are two current and important topics in software engineering [Agerfalk, 2006]. While Agile methods overcome well with increasing change business environments. So, there are many benefits from a fixable software engineering approach like Agile. So this study will help to understanding the challenges and from where they come.

Quality Function Deployment will give the organizations a quality paradigm to help them controlling the challenges and the obstacles.

1-5 Goals and Objectives:

The goal of this study is to gain understanding the challenges and the obstacles that occur in Large-Scale Agile Software Development, and putting some recommendations to overcome these obstacles, and after that it adopts quality paradigm “Quality Function Deployment” that surely will help in controlling the challenges and overcoming the obstacles faced upon adopting Agile methods at a large scale.

The objective of this work is to explore the weaknesses of using Agile methods in large-scale, that will give a better look and understood the better ways to avoid challenges and obstacles emanating from people, process, culture or combination of all. Moreover, a procedures and ways are recommended to avoid this challenges and obstacles and try not to face them.

This work will attempt to help the companies to take a piercing look into challenges to Agile in large-scale, and considering the proposed solutions and recommendations that attempts to overcome this challenges and obstacles. Flowing that, the new adopted quality paradigm (QFD matrix based for Agile large-scale software development) will help to control and reduce product and service development cycle times by as much as 75 percent with equally impressive improvements in measured customer satisfaction.

1-6 Methodology of the Proposed Solution:

In this work, the solution of research problem will be solved in two stages. Figure1.2 follows the recapitulation for each stage:

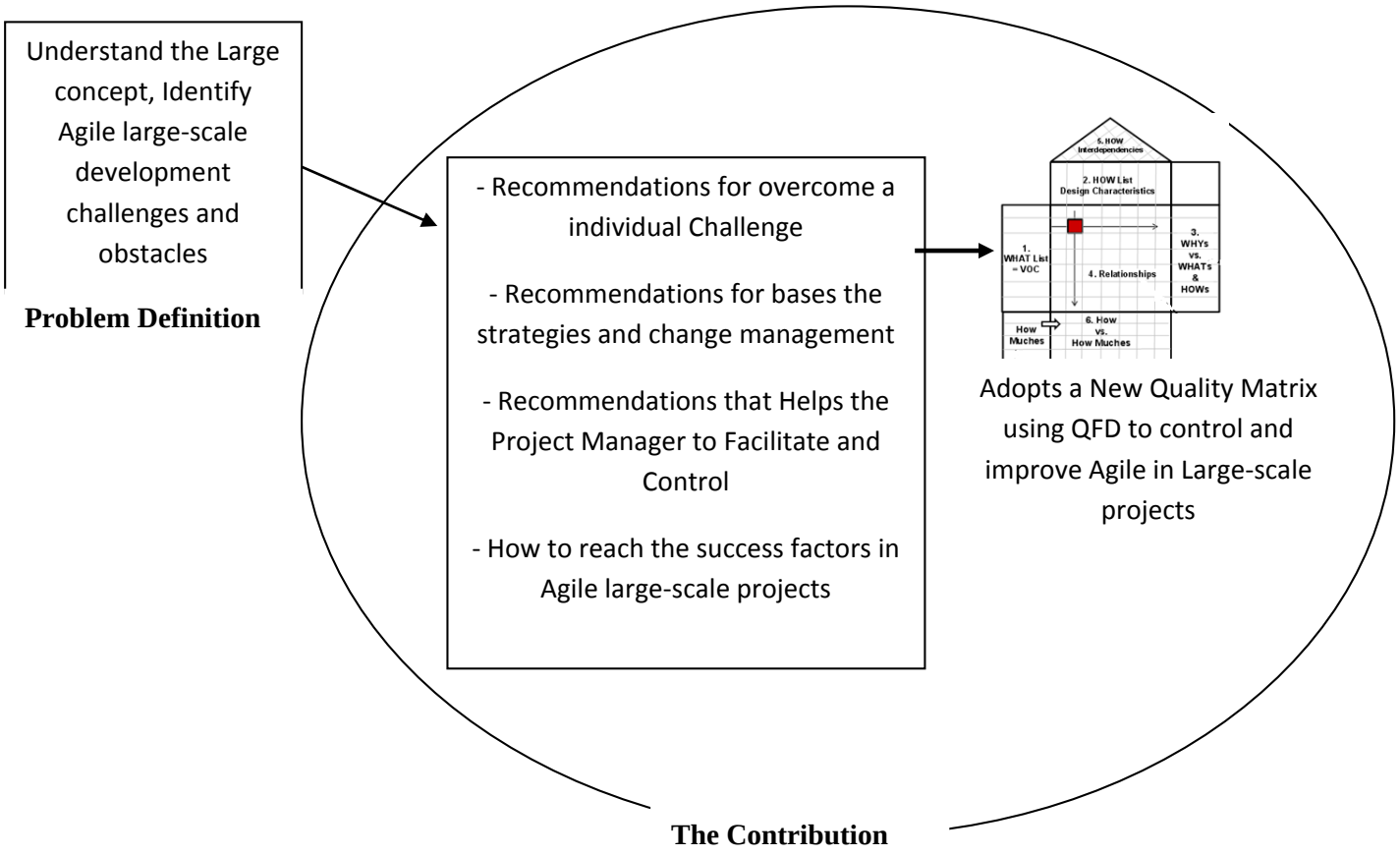


Figure 1.2: Methodology of the Proposed Solution

This study proposed solution that contains two stages; stage number one is to define the problem, and stage number two is the contribution of this study:

Stage 1 (Problem Definition): Finding out the differences between small and large scale, and why Agile has been worked well in small projects and not worked well in large projects. In order to reach the successes attributes, challenges and obstacles should be identified to prevent the large scale Agile software Development, and find out where they come from.

* **Stage 1 in Details:** In the past few years the researchers and the development organizations have concluded and identified many challenges in Agile large scale development process, some of them have reached that obstacles that come from people, such as lack of knowledge in development team, lack of self-management, communications between the team itself and the customer, team coordination and creation, project management, planning, documentation, fear to take a decision, lack of social skill, don't understand the principle of Agile methods, and others reasons that caused by people.

  The other group of researchers identified the obstacles that come from Agile processes itself, such as; requirements change or prioritizing requirements changes, Testing.

  The third group have identified the challenges that come from cultural and Policy quality procedures and standards in large organizations that been used and fallowed from long time.

In addition to, Agile methods blended technical and human behavioral together, so the challenges will come from both sides. Besides that, this study will consider another perspective; that there are challenges that come from environment change, business change, the variability of anything that affect on the project that allows a challenge to occur such as; people leaves their jobs, Natural disaster.

At the end of this stage, the study will give a clear and piercing look and fully list of possible challenges and obstacles that could face Scaling Agile in large Way.

Stage 2 (The Contribution)*: Proposition of recommendations that help to overcome the obstacles facing large scale Agile software development. Then using those recommendations helps to adopt a new quality paradigm (QFD) that transforms needs of customer-developer using the Agile methods into measurable objectives of design quality. Which help in controlling the challenges faced upon adopting Agile in large scale project.

* **Stage 2 in Details**:

1- After the stage1 fully identifies and understands the challenges that could probe in Scaling Agile to large projects, this stage will give some proposed solutions and recommendation that the development teams be able to overcome the challenges. Besides that, this study will propose a recommendation and procedure how to make a good platform to adopt Agile methods to large-scale projects.

2- This stage surly will help project management, to control the quality of the product, and to reach one of the most important goals of Agile methods; focusing on customer/stockholder value and needs *(Agile and QFD shared the same interested in this goal).* So for adopting Agile methods in QFD; Starting with the initial matrix, depicted in Figure1.3, the QFD methodology focuses on the most important product or service attributes or qualities.

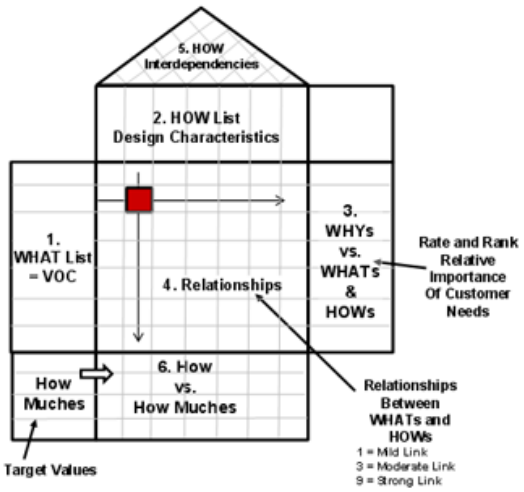


Figure1.3: QFD: 'House of Quality' Approach [ReVelle, 2004]

In Figure1.3, the definition of the terms:

1. What List (Voice of the Customer): Product specifications (Customer Needs)
2. How List: Process specifications; how that developer overcomes the challenges of Scaling Agile methods in large projects.
3. Why vs. What & How: Rating and ranking a relative importance of customer needs.
4. Relationships: it's the relationships between the product specifications and Process specifications.
5. How independencies: the relationships between one Process specification and the other of Process specifications.
6. How Much (Design Target): Determine target values for the design requirements.

Once you have prioritized the attributes and qualities (product specifications, Process specifications), QFD deploys them to the appropriate organizational function for action, as shown in Figure1.4. Thus, QFD is the deployment of stakeholders-driven qualities to the responsible functions of an organization.

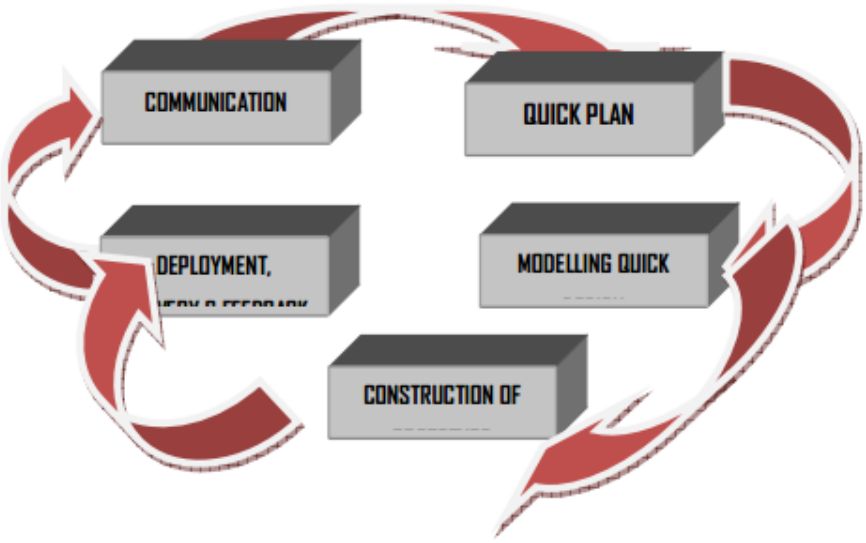


Figure1.4: The Prototyping Model QFD Matrices. [Okonta, 2013]

QFD practitioners claim that using QFD has enabled them to reduce their product and service development cycle times by as much as 75 percent with equally impressive improvements in measured customer satisfaction [ReVelle, 2004].

Chapter Two
Literatures Review

- Okonta O., Ojugo A., Raphael W., and Dele A. (2013)**, *Embedding Quality Function Deployment In Software Development: A Novel Approach,*** West African Journal of Industrial & Academic Research Vol.6, No.1, March, 50-64

This paper focuses on how customer's voice can be heard in order to reduce development costs and time, improve quality, and provide features that satisfy customer needs.

Furthermore, Quality Function Deployment has proven very successfully in producing products that appeal to customers; Quality Function Deployment provides linguistic continuity from customer to developers and brings organizations knowledge to bear on the product that achieves multifunctional approval. So this paper adopted a new approach to maintain quality in software development using the House of Quality Matrix.

The researchers concluded that QFD is a systematic mean of ensuring the customer requirements and needs that are precisely translated into relevant technical descriptors throughout stage of product development, so that will maintain and improve the product performance.

- Tenhunen T. (2010)**, *Challenging in Scaling Agile Software Development,*** Tampere University of Technology, Master Thesis

This research present many challenges that have been occurred when scaling Agile methodologies used in larger development projects. The team level practices like Scrum work poorly on the organizational level, but there are different strategies to get benefits of Agile to the whole enterprise.

Also it represents the communication within distributed teams that are still seen as a major challenge even though advanced technology is available. Face-to-face communication method is the most effective way of communicating and therefore teams should be co-located whenever possible. Being co-located, the team members will also benefit from the indirect osmotic communication, as most of the communication is being made inside the teams, global distribution can make a culture challenges which can prevent the success of Agile scaling.

Remark: The author mentioned that Agile adoption has become Mainstream; the organizations use practices from many different Agile methods and adopt them to fit the organization needs; so that it will take time or more than one project to adopt a fit Agile method for the organization.

- Conboy K., Coyle S. and Others (2011), ***People over process: key people challenges in Agile Development*****,** IEEE Software, July-Aug, 48-57

This published paper identified many serious people challenges experienced by large multinational organizations, all this organization using Agile over than three years. Furthermore, the researchers have many reasons to start this study; such as, the growing of using Agile methods and the boundaries of Agile are change; no longer restricted to small co-located teams and applied in environment outside their “comfort zone”, consequently presenting new people and HR management challenges.

The researchers of this paper have been defined nine challenges that affect scaling and using Agile methods, this challenge are; Developer fear caused by transparency of skill deficiencies, the need for developer to be a ‘master of all trades’, Increased reliance on social skills, A lack of business knowledge among developers, The need to understand and learn values and principles of Agile not just the practices, Lack of developer motivation to use Agile methods, Implications of devolved decision-making, The need for Agile -compliant performance evaluation, Lack of Agile -specific recruitment policies and suitably trained IT graduates.

Remark: Besides that, the authors have set of recommendations for each one of the challenges that they identified, and I think this is very important recommendations, but the only disadvantage of this paper that the researchers have identified the challenges came from people, disregards about the challenges came from environment and business, requirements, and prioritization changes.

* Drury M., Conboy K. and Power K. (2012)**, *Obstacles to decision making in Agile Software development***, The Journal of Systems and Software, Elsevier, 85, 1239-1254

This paper examines decisions made across four stages of the iteration cycle: Iteration Planning, Iteration Execution, Iteration Review and Iteration retrospective six illustrative mini cases were purposefully conducted as examples of the six obstacles identified in these focus groups. This included interviews with 18 individuals in Agile projects from five different organizations: a global consulting organization, a multinational communications company. Furthermore, this research concluded Agile software development teams are involved in critical decisions that underpin ultimate project success or failure. Because Agile software development teams exhibit characteristics that affect the nature of their team's decision making compared to traditional methods of software development.

The results indicate the six decision obstacles of decision making in Agile software development; unwillingness to commit to decisions; conflicting priorities; unstable resource availability; and lack of: implementation; ownership; empowerment. The researcher indicates; that these six obstacles make iteration execution and iteration review of Agile processes difficult as well, the people are unwilling to make decision they relaying on the scrum master, or on the person in charge, also the lack of the information prevents them from make their own decision making.

Remark: I think this research is have point that this six obstacles to decision making in Agile development will not affect the small project and the large project as well, I think no it will make more effect on large software development; because the decision must be faster and more sensitive than the decision in small software development.

* Sommerville Ian (2011)**, *Software Engineering, 9th edition*,** Pearson Education Inc., USA.

In Chapter three “Agile software development” the author explains the different between scaling Agile into small software system development and large software system development; came with six differences, such as, large system is brown field system (that is they include and interact with a number of existing systems), integration is important and necessary in large system, large system needs external rules and regulations that limiting the way that can be developed, and large system needs longer time so it’s make difficult to maintain bound teams who know about the system over that period as, inescapable, people move on to other jobs and projects.

On the other hand, the author put some difficult to introduce Agile methods into large companies for more than one reason, some of this challenges depend on people and the other depend on process (Agile process or organization procedure and culture).

In Chapter 23 the author has declare that Agile planning have two-stage approach, the first step is a Release planning; which looks ahead for several months and decides on the features that should be included in a release of a system. The second step is an Iteration planning; which has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2 to 4 weeks of work for the team. Nevertheless a major difficulty in Agile planning is that it is reliant on customer involvement and availability.

- Cao L. and Ramesh B. (2008), ***Agile Requirements Engineering Practices: An Empirical Study*****,** IEEE software, January/February, 60-67

This paper highlights on how rapidly changing business environment is challenging traditional requirement engineering, so when the development organization want to deal with requirement that tend to evolve quickly they need to use Agile methods, then the paper address the challenges that accrue when using Agile methods.

Furthermore, Cao paper had two phases. In the first phase, the study conducted cases studies in ten organizations involved in Agile or high-speed software development. That not followed any specific Agile methods. In the second phase, they collected data from six organizations that used XP, Scrum, or both. Then the researchers analysis the collected data and identified the benefits and the challenges of Agile requirement engineering.

This paper reveals that Agile requirement engineering differs from traditional requirement engineering in that it takes an iterative discovery approach. Agile development occurs in an environment where developing unambiguous and complete requirement specifications are impossible or even inappropriate. Another results is Agile requirement engineering is more dynamic and adaptive, Agile requirement engineering processes aren't centralized in one phase before development; they're evenly spread throughout development, Development organizations, should carefully compare the costs and benefits of Agile RE practices in their project environment.

- Shrivastava S. & Date H. (2010), ***Distributed Agile Software Development :A Review*****,** Journal of Computer Science and Engineering, Vol.1, 10-17

This paper discusses, there are many organizations begin to experiment with Distributed Software Development (DSD) facilities to solve challenges and problem in global development. DSD helped in reducing the cost involved and access to skilled sources. Sometimes, the search for competitive advantage forces organizations to search for external solutions in other countries, which is called Global Software Development.

Furthermore, there are benefits achieved by combining Agile with Distributed Development, seems to cause Increased visibility of project status and Agile process based on short continuous iterations make it easier to see the problems already on early stages of the project. Although, challenges faced Agile distributed teams, and this challenges and obstacles is occur in documentation, pair programming, training on Agile practices, different working hours, and distribution of the work.

The Authors have concluded distributed Agile development that it is possible to tap into new global markets and make best use of globally available talent, while potentially reducing costs. Also there are many benefits of using Agile methods with distributed software development. It helps in evaluating and measuring the progress of the project and problems of the project are more easily noticed at the early stage. It also handles the problems related to communication challenges in Global Software Development such as difficulties in initiations and maintenance of communication.

There are new challenges introduced when Agile is combined with distributed software development like communication misunderstandings and effect of different time zone, Increase in the amount of documentation also helps in achieving better coordination requirement clarification with use-cases and user-stories ,Agile teams work on user stories and not on component based tasks.

- Leffingwell D. (2007), ***Scaling Software Agility: Best Practices for Large Enterprises*****,** Addison-Wesley, Pearson Education Inc., USA.

In Chapter 8, the author has declared that the challenges of scaling Agile methods are two classes; the first, the challenges inherent in Agile itself, the second those challenges imposed by the enterprise.

The author set a number of challenges such as; lack of requirements analysis and documented specifications, culture and physical environment, team size, customer involvement architecture emerges. As a challenges come from Agile method that been used.

Moreover, Leffingwell identified another set of challenges that come from the enterprise itself; process and project management, policies and procedures, schedule, degree of distribution, and the relationship between developer departments and customer.

- Mahanti A. (2006), ***Challenges in Enterprise Adoption of Agile Methods: A Survey*****,** Journal of Computing and Information Technology, Vol.3, 197–206

This paper focus about introducing Agile practices into an organization an be a challenging task. And these challenges are Fear of change, office politics, closed mindedness, Black and white mind-set, outdated skills, Documentation-heavy mind-set, and Do-it-all-at-once attitude.

Furthermore, the researcher puts some strategies for successful adoption of Agile methods include: obtaining management buy-in, education and support to employees, integrating Agile practices to external processes, starting Agile pilot projects, reporting and adapting Agile project success, and sustenance of agility.

Also the author has puts some strategies for introducing practices in large organization, so he concluded that there is a challenges in introducing Agile to large projects; and defined the primary challenge is integration of Agile projects with the project environment's existing processes. Also mentions that there another challenges could occur such; large organizations usually distribute teams across several physical locations.

Also, he put determinants of change in large organizations; Obtain committed executive sponsorship, Articulate the 'burning platform' required for change, Form and maintain effective change team, Maintain long term focus, Adhere to key project and change management practices, Keep stakeholders engaged throughout the life-cycle, Align organization element, Empower affected employees in adopting the new system.

The author concluded that several challenges that have been inhibited the widespread adoption of Agile practices by the software industry. Understanding the dynamics of organizational change is a key to successful adoption of new methodologies. The success of Agile adoption is directly related to how the new methodologies are introduced in the organization.

- Nerur S. & Mahapatra R. (2005), ***Challenges of Migrating to Agile Methodologies*****,** Communications of the ACM 48(5), 73-78

This paper talks about challenges of migrating and starts using Agile methods, the paper discuss the effect of dynamic business environment has given elevation to emergent organizations that continually adapt their structures, strategies, and policies to suit the new environment. They need methodology that helps them to constantly evolve to meet the changing requirements and this methodology is Agile.

The authors discusses how Agile methods are perfect for projects that show high variability in tasks; changing requirements, in the capabilities of people, and in the technology being used.

Also they concluded that the biggest challenge here is to get the project manager to relinquish the authority he/she previously enjoyed. Also the selection of an appropriate method from the Agile methods that are currently available; that is one of the major challenges in Agile project management.

Chapter Three
Agile Software Development in Large Concept

Introduction

This Chapter will discuss two main subjects; the first is the differences between small and large scale Agile software Development, The second concerns with the success factors for global software development (Large-Scale Development) and how to incoming and reach this success factor and attributes.

The chapter is showing the differences between the small and large system development, and that lead to know why distributed /global / large-scale system not fully work under Agile methods process but in small-scale is effectively adopted and work under Agile process. Then, identifying of the success factor of distributed/ global/ large-scale development make to believe that it is can work in large project under Agile method.

3.1. Small vs. Large scale Agile software Development

Definitely, there is a big different between small system development and large development in a numbers of ways:

1- **Number of Distribution Teams**: Large systems are usually a collections of independent, communicating systems, where independent teams develop each system. Frequently, these teams are working in different places, sometimes in different time zones. It is practically impossible for each team to have a view of the whole system. Consequently, their priorities are usually to complete their part of the system without regard for wider systems issues [Sommerville, 2011].

2- ***'Brownfield Systems'****:* Large systems interact with a number of existing systems. Many of the system requirements are concerned with this interaction and so don't lend themselves to flexibility and incremental development. [Sommerville, 2011].

3- **Risk Analysis**: In Large systems risk analysis may be structured into more than one phase [Leveson, 1995]:

3.1. Preliminary risk analysis, where major risks from the system's environment are identified. These are independent from the technology that been used for system development. The aim of preliminary risk analysis is to develop an initial set of security and dependability requirements for the system.

3.2. Life-cycle risk analysis, that take place during system development and which is mostly concerned with risks that arise from system design decisions, different technologies and system architectures have their own associated risks. In this stage, you should extend the requirements to protect against these risks.

3.3. Operational risk analysis, which is concerned with the system user interface and risks from operator errors.

4- **Several Systems Involvement**: The process to integrate more than one system to create a system, a significant fraction of the development is concerned with system configuration rather than original code development. This is not necessarily compatible with incremental development and frequent system integration [Sommerville, 2011].

5- **External Rules and Regulation**: Large systems and their development processes are often constrained by external rules and regulations limiting the way that they can be developed, that require certain types of system documentation to be produced [Sommerville, 2011].

6- ***'Saturated'* State**: Lehman's fourth law suggests that most large programming projects work in a 'saturated' state. That is, a change to resources or staffing has imperceptible effects on the long-term evolution of the system [Lehman, 1996].

7- **Long Development Time**: Large systems need a long procurement and development time. It is difficult to maintain solid teams who know about the system over that period as, definitely, people move on to other jobs and projects [Sommerville, 2011].

8- **Tasks**: For a larger, more complex software project, a different task set would be required. It might encompass the following work tasks [Pressman, 2010]:

   8.1. Listed all stakeholders in the project.

   8.2. Interview each stakeholder separately to determine overall wants and needs.

   8.3. Build a preliminary list of functions and features based on stakeholder needs.

   8.4. Schedule a series of facilitated application specification meetings.

   8.5. Conduct meetings.

   8.6. Produce informal user scenarios as part of each meeting.

   8.7. Refine user scenarios based on stakeholder feedback.

   8.8. Build a revised list of stakeholder requirements.

   8.9. Use quality function deployment techniques to prioritize requirements.

   8.10. Package requirements so that they can be delivered incrementally.

   8.11. Note constraints and restrictions that will be placed on the system.

   8.12. Discuss methods for validating the system.

9- **Persons of interest**: Large systems usually have a diverse set of stakeholders [Sommerville, 2011].

10- **olitical Issues**: it can also be significant here and often the easiest solution to a problem is to change an existing system [Sommerville, 2011].

11- **Architecture:** Architecture in the large project is concerned with the architecture of complex enterprise systems that include other systems, programs, and program components. These enterprise systems are distributed over different computers, which may be owned and managed by different companies [Sommerville, 2011].

12- **Processes**: Large systems, the software design and implementation process is only one of a set of processes (requirements engineering, verification and validation, etc.) involved in software engineering [Sommerville, 2011].

3.2 Success Factor for Large-Scale Software Development:

While software is so important for all sides of the world, Agile software development itself is not a perfect process, so there are factors of success and on the other hand there are factors of failure and rather than that it's called "Challenges", so in this section will shadow on the factors of success.

3.2.1 Success factors for using Agile

According to Chow T. and Coa D. paper entitle *"A survey study of critical success factors in agile software projects"*, they put in a brief way the success factor upon using Agile Methods.

Dimension Factor	Success Factors
Organization	1.Strong executive support 2. Committed sponsor or manager 3. Cooperative organizational culture instead of hierarchal 4. Oral culture placing high value on face-to-face communication 5. Organizations where agile methodology is universally accepted 6. Collocation of the whole team 7. Facility with proper agile-style work environment 8. Reward system appropriate for agile
People	9. Team members with high competence and expertise 10. Team members with great motivation 11. Managers knowledgeable in agile process 12. Managers who have light-touch or adaptive management style 13. Coherent, self-organizing teamwork 14. Good customer relationship
Process	15. Following agile-oriented requirement management process 16. Following agile-oriented project management process 17. Following agile-oriented configuration management process 18. Strong communication focus with daily face-to-face meetings 19. Honoring regular working schedule – no overtime 20. Strong customer commitment and presence 21. Customer having full authority

Technical	22. Well-defined coding standards up front 23. Pursuing simple design 24. Rigorous refactoring activities 25. Right amount of documentation 26. Regular delivery of software 27. Delivering most important features first 28. Correct integration testing 29. Appropriate technical training to team
Project	30. Project nature being non-life-critical 31. Project type being of variable scope with emergent requirement 32. Projects with dynamic, accelerated schedule 33. Projects with small team 34. Projects with no multiple independent teams 35. Projects with up-front cost evaluation done 36. Projects with up-front risk analysis done

Table3.1: Success Factor for Used Agile Methods [Chow & Cao, 2008, p.963]

3.3. The Success Attribute of Agile large-scale Projects:

Beside the success factors, also there are a success attributes in the overall perceived level of success that Agile must achieved in the project:

1. **Quality**: Delivering good product.
2. **Scope**: Meeting all the requirements and objectives.
3. **Cost:** Delivering within estimated cost and effort.
4. **Time**: Delivering within estimated time.

Moreover, to reach the success attributes there are factors that the organization / developer must consider in a global / large-scale development projects:

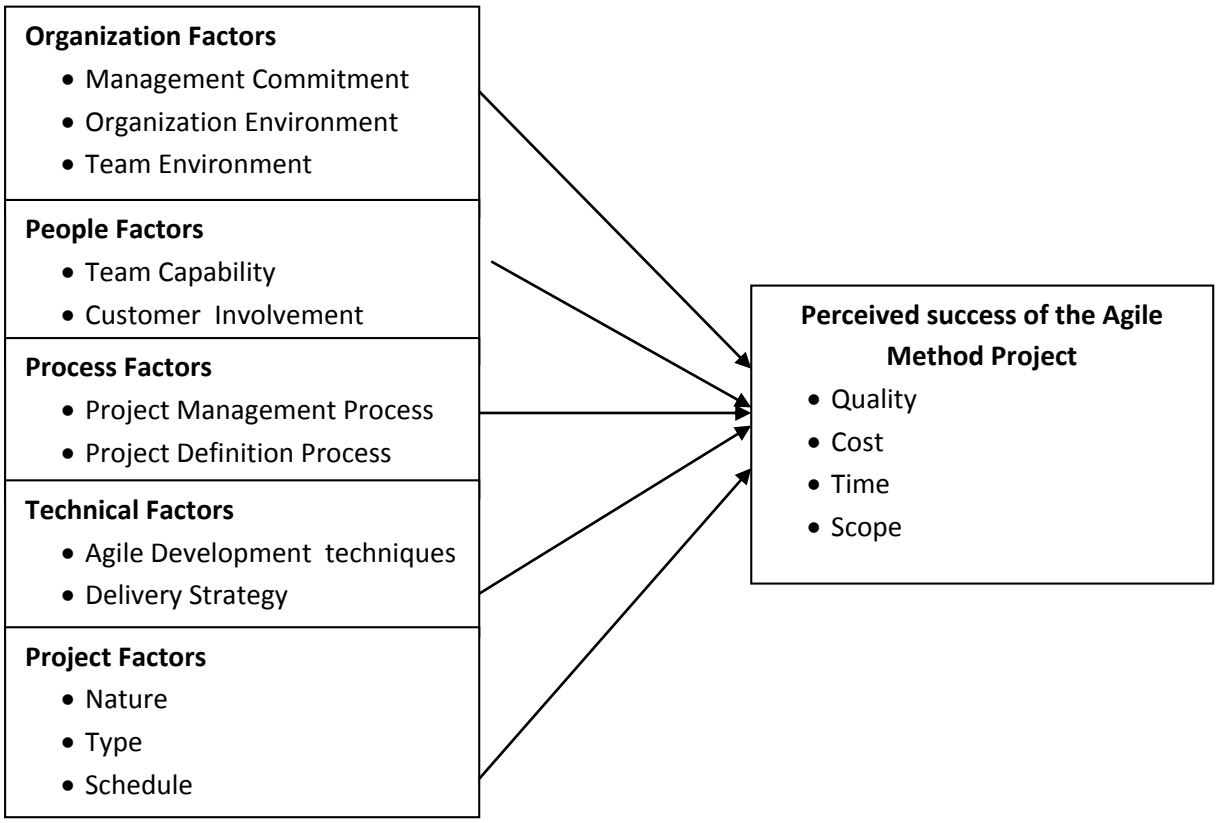


Figure3.1: Perceived success of The Agile Method Project [Chow & Cao, 2008, p.964]

Chapter Four
Challenges to Agile Large-Scale Development

Introduction

Since the beginning of the software development, challenges have been obstructing the way of delivered the Products. So software developers and engineers have been adopted and created software development methodologies to overcome this challenges and obstacles that occurring in the development processes. From waterfall model, Prototyping, Incremental development, Spiral development, Rapid development, Object-oriented development methodologies to Agile software development; this plurality of software development methodologies came from that every method solve a one or more of challenges that faced in development, or is ideal to solve a particular problem.

Since first emerged of Agile software development in the late 1990's, Agile has changed the landscape of software development; dealing with the rapidly changing business environment in which most organizations operate is challenging traditional software engineering approaches. Software development organizations often must deal with requirements, business domain that tend to evolve quickly and become obsolete even the project not complete yet [Cao & Ramesh, 2008].

So in the past few years the researchers and the development organizations have concluded and identified many challenges in Agile large scale development process, some of them have reached that obstacles come from people, such as lack of knowledge in development team, lack of self-management, communications between the team itself and customers, team coordination and creation, project management, planning, documentation, fear to take a decision, lack of social skill, don't understand the principle of Agile methods, and others reasons that caused by people

. The other group of researchers identified the obstacles came from Agile processes itself, such as requirements change or prioritizing requirements changes, Testing. The third group have identified the challenges come from cultural and policy quality procedures and standards in large organizations that been used and fallowed from long time.

Agile methods have blended technical and human behavioral together, so the challenges will come from both sides. also; that there are challenges that come from environment change, business change, the variability of anything that effect on the project that allows a challenge to occur such as; people leaves their jobs, natural disaster.

The following challenges are very effective. They could occur through adopting and using Agile software development methods and it's been divided in three groups:

4.1. The First Group: Challenges came from People, organization, and technologies:

4.1.1. Agile Planning:

Agile planning has two-stage approach, the first step is a Release planning; which looks ahead for several months and decides on the features that should be included in a release of a system. The second step is an Iteration planning; which has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2 to 4 weeks of work for the team [Sommerville, 2011].

Agile planning works well with small, stable teams that can get together (co-located) and examine the scenarios to be implemented. However, where teams are large and globally distributed, or when team membership changes frequently, it is practically impossible for everyone to be involved in the collaborative of planning that is major for agile project management. But the major challenges in Agile planning is that it is reliant on customer involvement and availability [Sommerville, 2011].

4.1.2. Knowledge Sharing:

Lack of business knowledge among developers, so they can't share any information about the project, and it's become more difficult to learn and share information about a large project specially the first sprint will deliver in 2-4 weeks, so the team they don't have the time to learn about the project domain [Conboy & Coyle, 2011].

4.1.3. Fear of Change:

Suspicion in people's minds that they wouldn't able to obtain the required Agile skills and overcome with the new agile environment may implant fear in their minds. Since fear of change is usually linked with a sense of loss, most people would refuse Agile introduction and try to undermine any related efforts [Mahanti, 2006].

4.1.4. The Integration of Agile Projects with the Project environment's existing process:

It's difficult to adopting Agile practices involves integrating each stockholder with the project environment's existing processes [Mahanti, 2006].

4.1.5. Lack of Necessary skill-set:

so this point involves many Sub-points:

A. lack of social skills: Agile practices such as co-located, on-site customer, meetings, and pair programming all need for social, communication and presentation skills. So it will be a challenge for some personnel were technically very talented but inherently weal in terms of communications and personal skill [Conboy & Coyle, 2011].So this challenge can distend and add more challenges for the project such as; bad customer relationship, bad team members relationships. That this will effect on the project construction and delivery.

B. Lack of team Work: In other worlds, team members can't do their job very well, and its go back for three possible reasons; first the team is not a good Agile team (that included team master, coder, tester, architect, customer, QA expert ... etc.) so the team will not have a good team work or not complete the work as required (its cannot be integrated). The second reason could back for bad choice or lack of the team members. The third reason is the master being a "jack of all trades, master of none" and not doing his/her job as "master of all trades" [Conboy & Coyle, 2011].

C. Lack of understand and learn values and principle of Agile methods, lack of motivation to use Agile methods, not just learn and understood the practices of Agile methods.

D. Documentation: Most Large organization favor Plan-driven approach where detailed sent and written to be constructed. But in Agile the teams tend to reduce documentation from the observation that a large part of documentation effort is wasted. So when teams are distributed there is a big chance to miss out some details about the project, so this is lead to suffering a cap in documentation that will make some ultimate misunderstanding for the project [Shrivastava & Date, 2010]. Also the Agile teams considers the documentation is a heavy mind-set process that worth to focus on, that's why they focus on the code and considered documentation is a "document as needed".

F. Closed and Old-Fashion Mindedness: so caused by the culture or the politics of team or / and organization is based on traditional development processes, that they can't see or understand Agile Method.

E. Wrong Selection an Appropriate Agile Method: this is one of the biggest challenges facing the project manager of Agile team [Nerur & Mahapatra, 2005]. The plurality of Agile method making this step is very risky for the whole project, that the project manager not select the appropriate method.

4.1.6. Distribution of Work:

it's a major challenge in large-scale development, is that work distribution should not take place according to the location. This will lead to an architecture which will start reflecting the team's geographical distribution. Different locations will become overspecialized in a particular component [Shrivastava & Date, 2010]. So it will become difficult to complete the user scenarios and stories within a iteration as different parts of the team will have to complete specific parts of the work on critical path to complete that stories.

4.1.7. Decision Making:

In Agile Software Development team in large organization deliver working software in short iterations, which results in more frequent, short term decisions, and in agile requires the team to adopt a collaborative and speedy decision making process [Moe & Aurum, 2012], so there are number of challenges in decision making in scaling agile in large development system [Dury & Conboy, 2012]:

- Agile team members are unwilling to commit to a decision and depended on the Scrum Master or others for decisions.
- Agile teams face opposite priorities for decisions.
- Decisions are based on unstable staff availability during an agile iteration.
- Agile team members are not implementing decisions.
- Agile team members are not taking ownership of decisions despite Agile Software Development team autonomy.

4.1.8. Risk Controls (Managements):

Agile development is a process of a short iteration cycles and provided a constant flow of code to the customer. Traditional risk management is a slow and comprehensive process. But Agile meant to simplify and deal with the project in flexible way, so the biggest problems when conducting risk management in Agile project environment are [Ylimannela, 2010):

- Resources are a problem. Since the goal is to create a working increment each sprint, it might be hard to find time and people to conduct risk management each sprint.
- Risk communication was a problem.
- Acceptance of residual risks. Who has the power to accept residual risk at different abstraction levels?
- There was no clear risk owner.

On the other Hand, there are many types of risk that could threaten the project in his life time, so these risks could be social, environmental, legal standpoint, economic, political, Technology risks. Then in large-scale projects the Probability of risk occurrence will be increase with the length of the project.

So Evaluating risk process is the best approach to determine how much agility is needed, for each project decision; consider the risk against little agility and the parallel risks of doing more planning and architecting [Boehm & Turner, 2005].

4.1.9. Existing Formalized Policies and Procedures and Corporate Culture:

There may be cultural reluctance to agile methods, especially in those organizations that have a long history of using conventional systems engineering processes. Or there are policies or procedures that have been used in waterfall-like practices and wont to adopt them on Agile Project.

4.1.10. Project management:

this challenge has involved a pattern called “Project Management challenge patterns” grouped into the six Declaration of Interdependence areas.

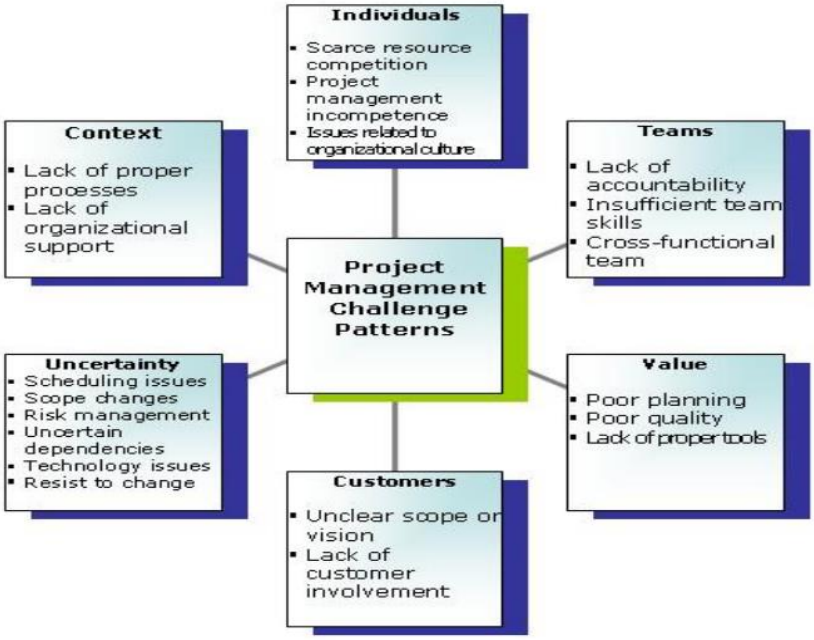


Figure4.1: Project Management challenge patterns [Sone, 2008, p.44]

4.2. The Second Group: Challenges come from The Method itself (Agile processes):

4.2.1. Testing:

Software testing in Agile development is very complex issue, Since Agile methods not focusing so much on Software testing activities and Agile development doesn't include testing practices required for a quality, So that make the process of finding the suitable testing method for large scale projects is challenging. Also the most challenge part in Agile testing that ensure that the solution of the problem in the development at the current iteration is done exactly.

On the other hand, there are two kind of testing manual and automated testing, so in Agile development it's hard to find a balance between the two kind [Koteska & Mishev, 2012].

Furthermore, continuous integration and changing requirements, it can be easy to miss critical tests for any requirement, also Code Broken Accidentally due to frequent builds will adding more challenges to test. Also the minimal documentation will add more difficult and challenges to test the final version against requirement and there is indefinite or unclear requirement the test will failed.

4.2.2. Location:

As traditional in Agile Methods the team must work in a co-located place, but in large-scale projects it's impossible to pool all 80-100 development member together so practically rather than creating a single developer team, the Agile encourages splitting developer into multiple smaller teams. But also this is not the only challenge that could face in location, most of the large-scale or global delivery the teams are distributed, so there is some sub-points involve:

A. Different Work Hours: This challenge comes form the distribution team witch spread over different time zone [Shrivastava & Date, 2010], so there is some unavailability of some team. So that's conflict with Agile method co-located characteristic.

B. Customer involvement: the customer involvement through time will reduce and that will lead to the situation called "hard maintaining stakeholder relationships".

C. Face-to-Face communication: this factor will also is a tradition in Agile so in distributed and large-scale will let it go so this will reduce the effectiveness of communication between customer and team. So the trust factor between them will Exposed to fail [Cao & Ramesh, 2008].

4.2.3. Communication:

The most effective form of communication actually is face to face communication. Agile teams try to get as much of it as possible: the whole team sits in one big room. They meet daily in a stand up meeting; they have daily direct contact to their customer. So earlier mentions customer involvement among time will reduce. Also Amicability is key factor in agile communication so if the team didn't familiar with each other's the communication factor will failed.

On the other hand the customers sometimes find difficulties to understand or trusting Agile method, so this will make more challenging in communication factor in the process of Agile method.

4.2.4. Requirement Engineering:

Agile involves prioritizing requirements in each development cycle. Prioritization often happens during the planning meetings at the beginning of each cycle. Also agile Requirement practitioners uniformly reported that their prioritization is based predominantly on one factor business value as the customer defines it.

But in Large Project it's could happened changing in prioritizing and the business value probably in very large way change. And it will affect all the previous work of the development.

There is more than one challenge in Requirement Engineering; at first cost and schedule estimation that based in user stories, the second accrue in minimal documentation when communication breakdown, the third when decided to identified nonfunctional requirements. So using business value that often focused on time-to-market, as the only or primary criterion for requirements prioritization might cause major problems; it's could lead unsalable system architecture [Cao & Ramesh, 2008].

4.2.5. Culture and Physical Environment:

In Agile effective environment look different than others; most team members work in open environment, and the process doesn't look very efficient because there is certain informality to stories and scenarios, so that seems team member do more talking than coding. So these scenarios may appear to be "unsupervised" or "unprofessional" because the team does not seem to afford the trim that come to expect in an efficient, well-organized enterprise, and that lead to "stuff stuck all over the wall" [Leffingwell, 2007].

4.2.6. Quality:

Agile teams have a quality challenge, When they "finish" their sprints and deliverables, but when they accumulate a significant amount of technical debt in the form of defects and poor design choices.

4.3. The Third Group: Change and Natural causes:

The Variability of Change; as been known change of any aspects of the project will effects on the cycle life of the iterations or the whole project life. So the there is a positive relationship (regularity) between the variability and the cycle (iteration) time (life). So that leads to increases the proportion of the emergence of challenges facing the system.

It must been consider two aspects here, the size of variability and the type of change. The first aspect the size could be large, moderate, and minimal. The second aspect is the type of change, such as; environment change, business change, or team change.

Agile software development, reduce the cost of change as shown in Figure4.2, but its cost is more in the early stage of development than traditional software development, so the team must be careful in the first stages of the development.

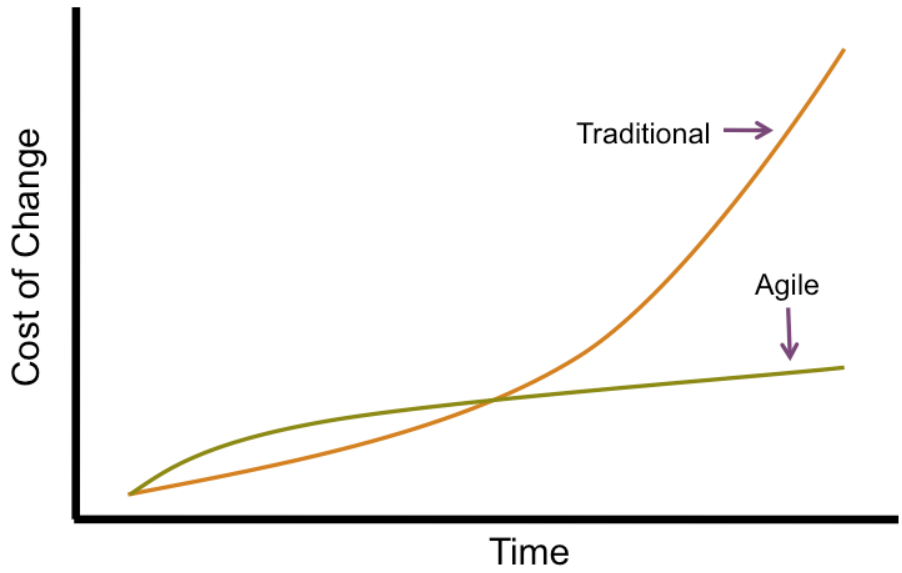


Figure4.2: Cost of Change [Pressman, 2010, p.68]

Change could come as result of forcibly circumstances, consequences for management faults, or natural disasters and causes.

Firstly, forcibly circumstances, for example an unprecedented technological developments; with the length of the project duration the organization may come up with an new technology that replaced the old one that has been determined the “business (environment) scope” in the large-scale project plan, especially the competition in the market between the major companies in the technology and communication sectors that release new and advance materials in a short period of time.

Secondly, consequences for management faults, the developers gain skills, experience and details through their working time in the project, that lead to more knowledge and experience in the project than others, and in large-scale projects that used Agile as development method they depend on team knowing the details rather than the documentation (minimal documentation: Agile basics), so when the project lost a team member that has the experience and the knowledge of the development processes and details by a management faults and let the team member leave the project without a good reasons that will lead to hinder the project.

Thirdly, natural disaster and causes, this concerns obstacles and hinders that force the development organization to delay the executing of the project, and that leads to losses and it might cancel the project.

Chapter Five
Agile Large-Scale Software Development and Quality Function Deployment (QFD)

Introduction

After identifying and understanding the challenges and the obstacles that probe in Agile large-scale software development, in the previous chapters. This chapter will present some suggestions and recommendations that attempts to help the use of Agile method effectively by development team and organization in the large-scale projects.

This recommended process assuredly helps the project managers to control and facilitate the process of scaling Agile in large-scale projects, some of those recommendation will try to help solving an individual challenge that been discusses in the previous chapter, the other recommended process will put inside the quality matrix using QFD matrix and call the process specification and its contain three aspects; how to reach the success factors in Agile large-scale projects, recommendations that helps the project manager to facilitate and control, and recommendations for bases the strategies and change management.

There are so many reasons behind why chose to combining Agile and QFD; Agile development has many shared properties with QFD, both trying to reach the highest degree of customer satisfaction and achieve what is really needed, focus on dealing seamlessly with the process of change and to avoid as much of the loss, and in addition to that the two methods are trying to reduce project time and reduce expenses and to increase the efficiency and the quality of the product to greatest form possible. Totally, these are the reasons why integrating agile with QFD.

Quality is one of the key factors in software industry. On the other hand, software systems have been developed as unique project in most cases, and produced in certain time limit. So, in Agile large-scale project it is really difficult to controls time, cost, quality, scope, and customer involvements, unless the team(s) been under control and quality assurance. Therefore as a software engineers we need to achieve massive tasks and delivered on certain time, provide features that satisfy customer needs, and reduce development effort and time.

As we know, human nature is the nature of *"not commitment*", especially in this study, the customer involvement will reduce during project time, and so, we need to focus on customer needs, to kept touch with the customer and knowing his thoughts.

So, when we use one of the Agile methodologies, we need to find an approach to compromise the views between the customer and the developers. Metaphorically, the customer speaks one language and developer speaks another. Therefore it's time to use an effective way to deal with the customer need and what really want "Customer voice", and that tool is the Quality Function Deployment that provides linguistic continuity from customer to developer;

5.1 QFD and Software Engineering:

QFD is given special importance to the customer needs, and adding values to the product through maximizing positive quality, QFD defines as a structures planning and decision making methodology, that capture customer needs and translating those needs into process and quality plan then into product.

Obviously, customer role is very important and significant in Agile software development, and in all software engineering, so there are a interest to drag QFD into the development field.

There are several attempts to integrate QFD and software processes, some of these attempts have been successfully using QFD as a tool to connect requirements within the action plan of an organization for its processes improvements [Liu et al., 2005].

The efficient use of QFD structured the possibilities for process improvement, which is very important in Agile large-scale project. The structure of QFD allows for an efficient reuse for the information that been gathered during the project. When the QFD process is repeated, most of the results from can be reused and merged with the new information that been gathered. Then, this will reduce the effort that been required for conducting the new iteration. [Hierholzer et al., 1998].

5.2 Recommendations for solve an individual Agile large-scale challenge:

There are many recommendations to solve an individual challenge to Agile large-scale software development that will help the project managers to overcome the most common challenges in Agile large scale development:

1- Knowledge Sharing:

- Training sessions on basics in the business domain and the customer company specific areas run by customer.
- Provide a small training module makes it interactive to allow developers gain position in the project business knowledge.
- Training sessions in IT and business knowledge.

2- Lack of Social Skills:

- Combine development and training program to provide customized training materials on social skills, and using developers' own examples.
- Using proper documentation to back up communication and details.

3- Lack of Team Work:

- Facilitating learning and distributing knowledge using pair programming and pair rotation.
- Encourage task self-assignment to allow developer work in different areas and learn new skills.
- Reintroduce specific roles when it is perceived useful to teams with e.g. large team size, conflicts between developers [Conboy & Coyle, 2011].

4- Lack of understand and learn values and principle of Agile methods, lack of motivation to use Agile methods:

- Ensure teams members to get Agile training.
- Agile coaching and championing.
- Ensure team's observation/validation of Agile practices.
- Assess agility in terms of Agile values not practice commitment.
- Collecting and sharing successful adoption stories and positive experiences of Agile methods.
- Hire the right developers and diversiform member.
- Use team recruiting to find right person working in the team.
- Put newly employee in Agile projects to get hands on experience.

5- Documentation:

- Maintaining valuable documentation may also improve large-scale development team collaboration process while using agile practices [Holmstrom & Fitzgerald, 2006].

- Providing user stories with use case diagrams in globally accessible backlogs helps to reduce misunderstandings and improve team collaboration [Hossain & Babar, 2009].

- Use various tools like issue tracker (e.g. Jira), project management tool (e.g. Scrum works) to helps in maintaining documentation and good transparency [Hossain & Babar, 2009].

6- Distribution of Work:

- Team should think about their work in the context of completing user stories not adding features to components.

- Teams need to consciously distribute tasks relating to a single story across the whole team, and think in terms of user stories not system components [Shrivastava & Date, 2010].

- Avoid breaking user stories into tasks and then assigning tasks according to geography and/or skill sets.

7- Decision Making:

Agile software development requires alignment of decisions on the strategic, tactical, and operational levels in order to overcome these challenges. Agile development also requires a transition from specialized skills to redundancy of functions and from rational to naturalistic decision-making. But still this takes time; the development companies needed from one to two years to change from traditional, hierarchical decision-making to shared decision-making in software development projects.

Also the company must consider these recommendations in the field of decision making:

- Build a sharing and learning environment to empower team decision-making.

- Use voting system in the decision that the team must make.

- Facilitator is the role that the project manager must act.

8- Testing:

The best solution of testing challenges to Agile development is using the "***Automated Agile Testing***" method that intended to minimize the quantity of manual work that involved in test execution, so gaining a coverage of a large number of test cases. So, this method will test developing environments that are usually unit tests that been responsible for quality of the smallest modules [Jureczko, 2008].

Automated Agile Testing requires automated tools, yet, the testers must keep involve, and keep in mind that these tools is not easy to learn. The selection of the tool(s) must be correct and meets the requirements, and depend on the project size, and the time allocated for the project costs. Also the testers must split the long test case into smaller ones that are easy to maintain [Arora, 2008].

9- Communication and Location:

Various tools have been suggested both formal and informal communication and project support, and especially for global and Geographical distance, so there are some of useful tools and recommendation to face the challenges that been reported in the previous chapter:

- Social networking tools: different social software tools and social networking tools encourage team's members to interact with each other and that will enhance the communication and social skills in the team(s).

- Communication tools: e-mails, messengers, video conferencing, and web cams.

- Software configuration management tools: repositories and version controlling tools.
- Bug and issue tracking tools: that contains the information about bugs found, and how to scrape and track bugs.
- Mail listing: make a list of all stockholders and any one related to the project and send the update and change to all.
- Knowledge centers: containing technical references and frequently asked questions.

10- Change and Natural Causes:

- Make daily backup for project and put this backup in various locations.
- Make sure that nobody looks to your agenda and give your secrets to a competitive company.
- Think about futuristic ideas, even if they are different, so you can control the market in the future.
- Don’t let go any valuable member of your team(s), because he/she may know details that other do not know.
- Don’t put one job for one person.
- Make a Contingency Plan in case there is a natural disaster or causes occurred.

5.3. QFD House of Quality Matrix based for Agile Large-Scale Software Development:

5.2.1- What List (Voice of the Customer):

In this list, it must contain all things that related to the customers' needs and what they really wants, despite of the developer is using Agile or any other development methodology. The only thing it should be taken under considerations is the large-scale.

What List (Voice of the Customer)	Explanation
Ensure quality of products	Delivering good product
Increase Customer satisfaction	Delivering within estimated time
Easy to implement and use	
Be practical product	
Be a scalable* product	System, whose performance improves after adding hardware or software, proportionally to the capacity added, is said to be a scalable system.
Reduce Costs / return the investment	Delivering within estimated cost and effort.
Flexibility to address operational challenges	
Control / visibility into process:	Visibility and control are important so that you can ensure that everybody is rowing together toward success.
Improve operational efficiency and effectiveness	
Build up a credential to stand against competition	
Meeting all the requirements and objectives	Scope

Table5.1: Customer voice “what” list

5.3.2 - How List (Process specifications):

In this list, it must contain all related things to overcome the challenges and the obstacles in Agile large-scale software development, to ensure the success to the project.

How List (Process specifications)	Target
Reduce Ambiguity	↑
Maximize Stability	↑
Understand Dependencies	↑
Facilitate Coordination	↑
Balance Flexibility and Rigidity	o
Managing Risks Risks Identification Mitigating Risks Monitoring Risks	↑

Quality Assurance Measuring Process Quality Increased Communications Put a delivering scope during each sprint Morale of the developers Measuring product Quality Product Maintenance Quality Fix Backlog Fix response time Delinquent fixes Fix quality	↑
Infrastructure supported for large-scale software development Criteria for Selecting Infrastructure Communication and Collaboration Infrastructure Mailing list for Emails Infrastructure for weekly meeting Forums for discussions, queries, change, and defect tracking Processes to facilitate large-scale development infrastructure Will-Defining Task Exclusive Areas of Responsibility	↑
Requirements Change Management Impact Analysis Updating the associated artifacts Preplanning	↑
Education and support to employee with Agile Values	↑

Integration Agile practices to external processes, and observation and validation of Agile practices	↑
Change Management Gain committed executive sponsorship Explain the 'burning platform' required for change Form and maintain effective change team Adhere to key project and change management practices Keep stakeholders engaged throughout the life-cycle Maintain long term focus	↑

Aim to Maximize ↑

Aim to Balance ○

Table5.2: How List (Process specifications) list

5.4. Define the Process Specifications for Agile Large-Scale software Development:

1- How to reach the success factors in Agile large-scale projects:

To reach the factors of successes, there are many issues that the organization / developer must put in their considerations when starting develops a global / large-scale development projects:

Reduce Ambiguity: Large-scale and global projects do not have the same "luxury" as the small project, so ambiguity leads to assumptions, and these assumptions are Not be easily visible, so that leads to conflicting assumptions that not be exist for some time before appear as a problem. So that problem leads to preplanning, redesigning, reconstruction, and rework. So this set a difficult to implement in a distributed and global environment. Nevertheless, ambiguities can lead to confusion in the organizational processes, or with management practices, requirement, or / and the design.

So if this combined with lack of experience and knowledge in the development teams, the status get even more and more complex. What is clear expression to team with particular background, culture, and domain knowledge may not well clear to another. So that required techniques for verifying the aspects of the projects understood. So the team should balance technology aspects and domain knowledge.

Maximize Stability: Instability of any project will have an influence on many aspects. As we know Agile processes are response for unclear and unstable requirements in other word “Dynamic” requirements. So Agility try to create an environment that optimizes the amount of ad hoc and informal communication working in a collected space, co-located customer and team, short iterations, daily meetings…..etc. but in distributed and global projects where this communication are difficult, Stability is a factor. So the change request of any project especially large one is a big risk, in large projects needed 2.4 times longer [Herbsleb & Mockus, 2003] to resolve in the global/large-scale environment compare to co-located project. So in Agile we need for excellent requirement engineering and architecture. So that means that there are more prototyping, designing, frameworks to developed, and development environment customized, and other ways to increase stability of the aspects of the projects.

Understand Dependencies: The understanding of interrelated nature of the tasks, coordination, volume, and the frequency needed among the global/large-scale projects. This process is critical for planning and the execution of the projects. So there is relationship between subsystems of the projects called “dependency”, so the coordinator of the project must deal, understand, identify and learn about these dependencies, so he/she/they can maintain these relationships.

Facilitate Coordination: The understanding of interrelated nature of the tasks is to facilitate the corresponding coordination. So coordinate leads the team to be able to do what is proportional with the needs. Furthermore, there are many ways that allow teams to coordinate via communication, management practices, processes, and product line architecture. In global/large projects the coordinator must be able to find a ways to maintain Coordination, for example in large project organization want to achieve cost reduction by moving some of their development to low-cost sites, but Maintain the quality of the product.

Balance Flexibility and Rigidity: Large-scale and global projects should be more flexible and more rigid than co-located (small-scale) projects. The overall development process must be flexible to fit the differences between the teams (their knowledge levels, skills, cultures, and practices). But it must be rigid enough to ensure the practical aspects of the projects are well defined and followed as the normal, such as instructions are understood, architectures are complied with, requirements are achieved, configuration management processes are defined, integrations and test procedures are appropriate…. etc. So this necessary to monitor progress, and ensure deadlines and quality.

2- Recommendations that Helps the Project Manager to Facilitate and Control:

So, there are many aspects that the team(s) and the organization (development and customer) should follow that attempts to overcome the challenges and the obstacles that face upon Agile large-scale software development.

Managing Risks: this aspect contains three operations; risks identification, mitigating risks, and monitoring risks.

1. Risks Identification: as well as in traditional software development projects there are several methods and ways to identify risks that also valid in large-scale projects such as Software Risk Evaluation (SRE) and ATAM, also there other ways to identify risks by reporting from teams member, through periodic management meeting, or by voluntary submission. However, in the process of identify risks the team finds the degree of risk to schedule, budget, and quality depend on the system that has been developed, the context that will be developed, and the characteristics of the team [Sangwan et al., 2007] So to deal with this aspect the development organization must determine the coordination capabilities, and understanding the risk inherent in a project to get some insight into the organization and look at the option for the system.

2. Mitigating Risks: There are areas to be considered when mitigating risks; the first is to looking at ways to improve and increase the development organization's capabilities. The second is to plan the coordination aspects of the project, and the third is to pre-plan how to going to adjust the project in the event where the original plan is not enough.

3. Monitoring Risks: monitoring risk in the term of large-scale is quite difficult. It's difficult to adequately understand what is really happening within the teams or geographically distributed. So, through time the deliverable sprints (versions) are late or quality issues discovered, so the schedule has suffered substantially. It takes more time and effort to recover from this situation.

Quality Assurance: QA in large-scale is different from small and co-located projects. When the problem discovered in large-scale and global projects its get more difficult to recover from it, to know who to hand off the problem to, and to know who are all the parties who should involve to solve the problem. Depending on the type of issue that takes place (e.g., requirements misunderstanding, syntactic and semantic issues...Etc), maybe it will discovered at different times. So there are some operations that must be considered such as measuring process quality, measuring product quality, and Product Maintenance Quality.

1. Measuring Process Quality: Obviously, coordination and control is more difficult in large-scale projects, so we want to put processes in a place that will ensure that delivered products of high quality and ensuring that quality doesn't affect the other three attributes of successes (time, cost, and scope). So, there are some recommendations that the development enterprise must take in their considerations:

1. *Increased Communications*: this means start using the phone, fax, e-mail frequently, using face to face meetings just between team's masters, or using the social network and video conferences.

2. *Put a delivering scope during each sprint*: Because of Agile nature of planning process, it's normal to have the scope for each sprint to avoid many situations that could involve in each sprint such as if the team delivers less than what is expected.

3. *Morale of the developers*: the development enterprise should always give the developers credit and thankful of their work, so that leads to make the developers feels comfortable and secure, so they will give more intention and loyalty for the work and the project.

2. Measuring product Quality: product quality measured by tracking the counts of defects identified during the incremental testing of the previous releases / sprints [Paulish, 2002]. The defect counts obtained from change management tool where it's been able to summarize the priorities, counts, and status of defects as they have been reported by the test team. So, in large-scale it gets more difficult to track the semantic errors, misunderstood requirements, and the level of understanding of any aspects. That makes the large-scale problem which the researchers recommended better practices such as automated code inception, using tools check for violations of coding standards, and using some practices of testing:

* The central team enumerates acceptance test cases derived from the high-level functional requirements.

* The remote teams create automated test classes as part for each sprint for their respective task assignments.

* The central team specifies and executes integration tests at the central site.

3. Product Maintenance Quality: Product maintenance quality is measured by the effectiveness and efficiency of the maintenance process. Typical measures founded to assess the quality include [Kan, 2003]:

1 *Fix Backlog*: a measure for the rate at which fixes for reported defects become available.

.2 *Fix response time*: a measure of the mean time of all defects from the time they were reported to the time they were fixed.

3 *Delinquent fixes:* a measure of those fixes for which the turnaround time exceeds the required response time.

4 *Fix quality:* a measure of bad fixes, that is, fixes that do not correct the defect or inject further defects.

Infrastructure supported for large-scale software development: there are some strategies and tools for infrastructure management in large-scale / global software development environments, Related to this point, three aspects will be discussed; criteria for selecting infrastructure, infrastructure required for communication and coordination, and processes to facilitate the usage of infrastructure appropriate and effective way.

1. Criteria for Selecting Infrastructure: In the process activities, making a proper project infrastructure for large-scale and distributed development is a significant project success factor. So, infrastructure tools and strategies must support collaboration and concurrency, processes, accessibility, and awareness and integration. Traditional tools that are used for small and co-located development may not fit the infrastructure requirement for large and global projects and that will be costly consuming and time consuming.

2. Communication and Collaboration Infrastructure: this infrastructure should be designed carefully, so that it supports the strategy and it is a significant goal. So there are effective communication infrastructures that the team must build:

    2.1 *Mailing list for Emails*: must send all works and updates to all team member and customer mailing list.

    2.2 *Infrastructure for weekly meeting:* it should be a weekly management meeting which involve all sites (teams, management, and customer). To facilitate this, video conferencing, voice chat, desktop sharing technologies, telephones, and social network utility must be used.

2.3 *Forums for discussions, queries, change, and defect tracking:* That helps the tester and the management to know details about any things happened in the life cycle of the sprint or the project without going back to the developers.

3. Processes to facilitate large-scale development infrastructure:

Infrastructure could not meet its objective without being attached to a process that is meaningfully defined and accurately executed. There are several aspects that can be taken into consideration here:

3.1 *Will-Defining Task*: This will increase awareness in the sense that each person knows what the others should work .That is an important aspect to collaborations with long distances, and reducing the misunderstandings between the teams.

3.2 *Exclusive Areas of Responsibility*: it is a technique to allocate (e.g. a set of files or modules). During new development it is often possible by having a suitable product structure and to split the responsibility between developers, So that it is important in large-scale to achieve clear tasks, well-defined interfaces, increase the awareness, look to enlarge connection between developer in different locations, give a clear seeing on a daily basis work and progress rather than meeting [Sangwan et al., 2007]

Requirements Change Management: Changing management is a difficult area under the best circumstances. We talk about managing the wave effect of requirements change, so managing the change includes:

- *Impact Analysis*: what is the cost of change? It is a must to understand the activities that involved.

-*Updating the associated artifacts*: both requirement and associated design must be updated.

-*Preplanning*: if change been accepted, there is a need to make modifications into the project schedule.

3- Recommendations for bases the strategies and change management:

On the other hand, there are some challenges that could occur in the life cycle of the projects within using Agile methods under large-scale projects, so we will be suggest recommendations for some challenges and obstacles that we talk about in the previous chapter.

At first there are some strategies for successful adoption of Agile methods:

1- Chose the right project.
2- Education and support to employee
3- Integration Agile practices to external processes.
4- Starting Agile project, reporting and adapting Agile project success, and sustenance of agility.

Moreover, the development enterprise must take into consideration the change principle in large-scale projects [Mahanti, 2006]:

1- *Gain committed executive sponsorship*: in large-scale adoption active sponsorship must be gained from the management, so its important for change initiative.

2- *Explain the 'burning platform' required for change*: the meaning of burning platform, putting a scenario where someone is forced to make a decision.

3- *Form and maintain effective change team*: Having effective project teams is essential for organization. The team should be characterized by brilliance, cohesion, good communication skills, correct decision-making skills, and accountability.

4- *Adhere to key project and change management practices*: Monitoring project metrics, managing project scope, and understanding dynamics of change can design an effective change methodology.

5- *Keep stakeholders engaged throughout the life-cycle*.

6- *Empower affected employees in adopting the new system*: Providing training and support to affect employees can reduce confusion, fear, and speculation.

7- *Maintain long term focus*: Should maintain long term focus to preserve agility in the organization.

5.5. Composing what list and how list in the House of Quality Matrix:

The construction of the QFD house of quality matrix is complete by placing the customer needs list as rows and the how list (process specifications) as columns of the matrix. And the relationship between rows and columns measured in one of four levels: strong, moderate, weak, or non-existent, and with values (9,3,1,0). The relationship between the rows and the columns is called correlation and it is determined by asking how significant is process specification X in satisfying customer need Y, and this correlation could be a negative which means that is improving a process specification prevents to accomplish the corresponding customer need.Those relationships have a weight called “importance weight” of the process specifications is calculated by multiplying the correlation value of each cell with its relative weight and adding (sum) the result of each individual cell in the column, The negative weight not involved in the calculation, so the importance weight of the process specification means how important those specifications are to be controlled.

*Importance weight of process specification = ∑ (correlation value * relative weight)*

*Relative weight of process specification = Importance weight / ∑ (Importance weight)*100*

Figure5.1: a New QFD Matrix for Agile large-scale software Development (Extract)

5.6. QFD as a method to improve Agile large-scale software development:

The aim of using QFD in this study was to clarify what kind of benefits the use of QFD offers during the Scaling the Agile in a large development project. So, QFS has the capability hopes to reduce total life cycle costs, improve software product quality, reduce evolution timelines, and meeting all requirements and objectives.

QFD is given a way to reduce the project manager efforts and time, the reason is QFD matrix is achieves and clear all needs and clear the way how to deliver those needs.

So, the QFD gives solutions for a number of problems that could occur in any software project especially in Agile project:

- *The Software Crisis*: This crisis is caused by the difficulty in and the failure of successfully evolving complex or large-scale software systems.
- *Increased Demand for Quality Software*: Software demand is increasing at an astounding rate; industry has found it difficult to find the necessary professional talent required for meeting this software development demand. As been reports in this study or others that Agile focused in quality, so QFD is represent away to ensure quality product.
- *Inappropriate Customer Developer relationship and Communication*: better communication between software customers and developers is needed. This has been a huge challenge in agile development especially in large-scale project.
- *Lack of High Assurance Systems*: Software industry requires ever increasing numbers of high-assurance software systems, particularly in mission-critical activities, business financial transactions, and life critical medical applications.

The QFD matrix represents a systematic means of ensuring that customer requirements are accurately translated into relevant technical descriptors throughout each stage of product development. QFD provides open, modular software architecture for future improvements. The flexibility of this tool providing an integral framework that allows software engineers to interface with existing and future software development tools and models.

Besides the above, QFD structure allows ruse of the information gathered during the project, as using Agile methodology repeats every sprint in a iterative way, we collected new information and easily can reuse the old information from the previous sprints and merged with the new information gathered, thus greatly reducing the effort required for conducting the new sprint. Yet, the use of QFD efficiently structured the long-standing discussion about possibilities for process improvement.

QFD allowed for condensation of the improvement effort on the most important problems with the Agile development current process. So this study adopted new (Figure5.2) enhancements in the process of improvements the product using Shewhart-Deming Cycle (Pane, Do, Check, and Act) that used as the base of incremental improvement for Toyota (TPS), and this cycle is considered as fundamental basis of incremental or Agile software development; because the key element in the cycle is the notion of repetitive adjustment in the process or product, based on the feedback obtained from the customer.

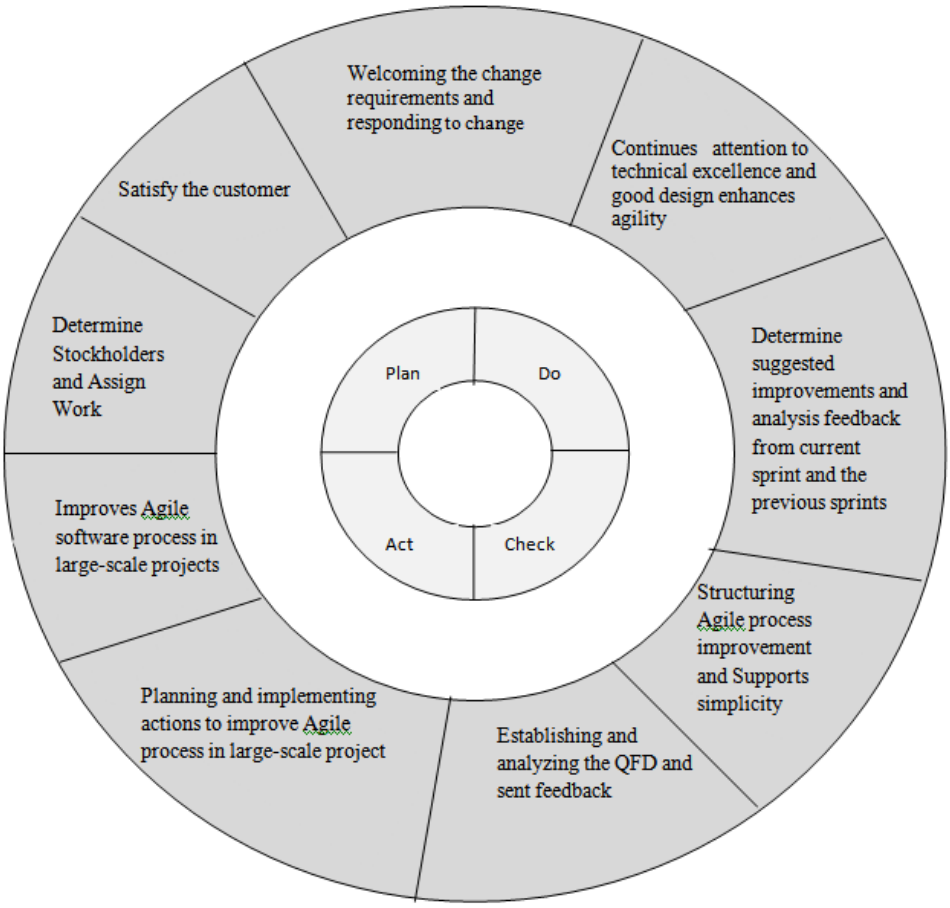


Figure5.2: QFD to improve Agile large-scale process using Deming's Plan-Do-Check-Act cycle.

5.7. QFD to improve the process workflow in Agile Large-Scale Development:

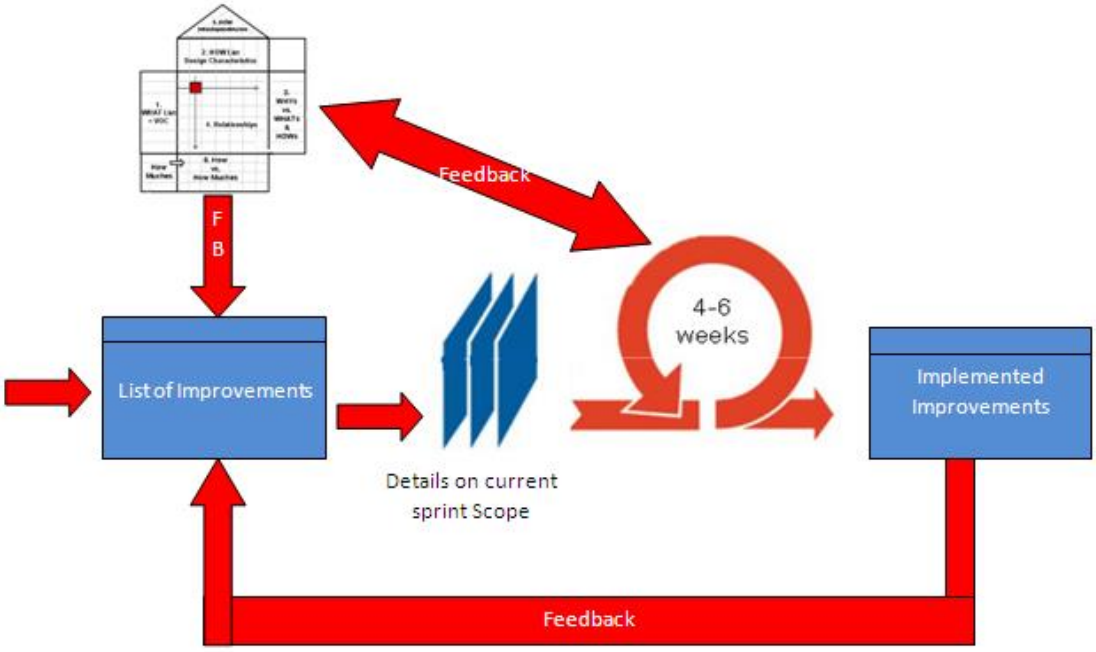


Figure5.3: QFD improve the process workflow in Agile Large-Scale Development

So the new enhanced cycles in Figure 5.3 enable the organization to:

1- Identify and resolve issues early.

2- Learn how the improvements work and how to tackle difficulties.

3- Adapt to changing business needs.

4- Respond to feedback and lessons learned

5- Evolve and enhance quality.

6- Involve the customer in all process (Make sure to involve the customer is an Agile principle).

7- Respond to changes in early stage.

Chapter Six
Conclusion and Future Work

6.1 Conclusion:

This study addresses three of the most important topics in software engineering today, Agile development, large-scale projects, and quality and process improvements. It will give a chance not just to overcome the challenges that faced upon using Agile methods in a large-scale projects, but gives solutions, bases, and infrastructure methods to facilitate, control, reduce, and organize the work under Agile large-scale projects.

The research contains three main chapters (3,4,5), each chapter has a purpose; chapter Three and Four is to define the problem in Agile large-scale development, chapter Five is the contribution of this study and help to establishes a successful factors for a well-adoption of Agile method in a large-scale projects:

1- Chapter three "*Agile Software Development in Large Concept*": This chapter importance show the different between scaling small and large projects, what is the success attributes for large-scale and how to reach this attributes.

2- Chapter four "*Challenges to Agile Large-Scale Development*": This chapter giving a piercing look into all challenges and obstacles that could been face upon using Agile in large-scale projects, this thesis reported over twenty five (25) challenges and where this challenges came from, where and when could happened.

3- Chapter Five "*Agile Large-Scale Software Development and Quality unction Deployment (QFD)*": This chapter attempts to help to control and reduce the software product under Agile large-scale methods, also to help the project manager to deal how to reach the successes factors that is critical operations and very useful to the business case and needs when projects with several teams are geographically distributed and across multiple time zones, this operations is meant to encompass the fully-experience to the project processes, teams, structures, and the system to be built, the other step is using the recommended procedures and processes to control the project using a new quality paradigm "QFD" as a method to improve the adoption of Agile, and improve the process workflow in Agile Large-Scale Development.

At the end of this study that hope to explore the weaknesses, solve those challenges and improve the using of Agile under a large-scale projects to reduce the product and services development time and efforts. So the using of the adopted QFD will assure to transform the requirements and the needs of the customer into measurable objectives that's helps to controlling the production developments and that will assuredly help and reduce the time of the project managers and give them a solution to overcome the challenges and obstacles in Agile large-scale and gives them method to establish a well-adoption for Agile in Large-scale projects.

This study has been achieved all goals and objectives and solve all the problem statements questions, as the researcher gained a better understanding of the relationship between the processes of Agile large-scale development methods and puts several suggestions to enhance the quality and ensure the adoption of Agile methods in large-projects even before the teams starts to develop and during the development cycle.

6.2 Future Work:

There are some recommendations for future work in this field based on the results of this study:

1- Put the new Quality of House matrix base for Agile large-scale software development in implementation in a real life project.

2- Try to use the CASE tools (Computer-Aided Software Engineering) to solve some problems that faced in the using of Agile method in large-scale projects.

3- Try to use others Quality Control and Assurance, and Value Engineering methods to enhance scaling of Agile methods, such as using the Modular Function Deployment (MFD) that's considers QFD is the first step and others steps are involve for company-supportive product modularization.

References

1. Agerfalk P. (2006), ***Towards Better Understanding of Agile Values in Global Software Development***, Conference on Advance Information System Engineering, Namur University Press, 363-374.

2. Akao, Y., ed. (1990), ***Quality Function Deployment***, Productivity Press, Cambridge MA.

3. Ambler S. (2006), ***Supersize Me***, Dr. Dobb's Journal, February.

4. Arora A. (2008), ***Agile Automation Testing***, Adobe Systems India Pvt. Ltd, Noida.

5. Boehm B., Turner R. (2005), ***Management Challenges to Implementing Agile Processes in Traditional Development Organizations***, IEEE Software, Sep-Oct, 30-39

6. Cao L. and Ramesh B. (2008), ***Agile Requirements Engineering Practices: an Empirical Study***, IEEE software, January/February, 60-67.

7. Chow T. & Cao D. (2008), ***A Survey Study of Critical Success Factors in Agile Software Projects,*** The Journal of Systems and Software 81, 961–97.

8. Clausing D. (1994), ***Total Quality Development***, ASME Press, New York, USA.

9. Conboy K., Coyle S. and Others (2011), ***People over process: key people challenges in Agile Development***, IEEE Software, July-Aug, 48-57.

10. Drury M., Conboy K. and Power K. (2012), ***Obstacles to decision making in Agile software development***, The Journal of Systems and Software, Elsevier, 85, 1239-1254.

11. Herbsleb D. & Mockus A. (2003), ***An empirical study of speed and communication in globally distributed software development***, IEEE Transactions on Software Engineering, 29(6), June, 481–494.

12. Hierholzer A., Herzwurm G. and Schlang H. (1998), ***Applying QFD for Software Process Improvement at SAP AG***, Conference of the World Innovation and Strategy Conference in Sydney, Australia, August, 85-95.

13. Holmstrom H., Fitzgerald B., and Others (2006), ***Agile Practices Reduce Distance in Global Software Development***, Information Systems Management, summer, pp. 7- 26.

14. Hossain E., Babar M, and Paik H. (2009), ***Risk Identification and Mitigation for Using Scrum in Global Software Development: A Conceptual Framework***, 16th Asia-Pacific Software Engineering Conference.

15. Jureczko M. (2008), ***The Level of Agility in Testing Process in a Large Scale Financial Software Project***. Software Engineering Techniques in Progress, University of Technology, Poland.

16. Kan S. (2003), Metrics ***and Models in Software Quality Engineering, 2nd edition***, Addison-Wesley, Boston, MA.

17. Koteska B, and Mishev A. (2012), ***Agile Software Testing Technologies in a Large Scale Project***, BCI Local, volume 920 of CEUR Workshop Proceedings, 121-124

18. Leffingwell D. (2007), ***Scaling Software Agility: Best Practices for Large Enterprises,*** Addison-Wesley, Pearson Education Inc., USA.

19. Lehman, M. (1996). ***Laws of Software Evolution Revisited***, Proceedings of European Workshop on Software Process Technology (EWSPT'96), Springer-Verlag. 108–24.

20. Leveson, N. (1995), ***Safeware: System Safety and Computers***, Addison-Wesley Professional, USA

21. Liu F., Sun X. and Kane G. (2005), ***QFD Application in Software Process Management and Improvement Based on CMM***, International Conference on Software Engineering, 1-6.

22. Lockamy A. and Khurana A. (1995), ***Quality Function Deployment: Total Quality Management for New Product Design***, International Journal Quality and Reliability Management, Universal Press Ltd. (UK).

23. Mahanti A. (2006), ***Challenges in Enterprise Adoption of Agile Methods: A Survey***, Journal of Computing and Information Technology, 3, 197–206.

24. Moe N., Aurum A. & Dybå T (2012), ***Challenges of Shared Decision-Making: A Multiple Case Study of Agile Software Development***, Journal of Information and Software Technology, 54, 853–865

25. Nerur, S. and Mahapatra R. (2005), ***Challenges of Migrating to Agile Methodologies***, Communications of the ACM 48(5), 73-78.

26. Okonta O., Ojugo A., Raphael W., and Dele A. (2013), ***Embedding Quality Function Deployment In Software Development: A Novel Approach***, West African Journal of Industrial & Academic Research Vol.6 No.1 March, 50-64

27. Paulish D. (2002), ***Architecture-Centric Software Project Management***, Addison-Wesley, Boston, MA.

28. Pressman R. (2010) ***, Software Engineering: A Practitioner's Approach, 7th edition,*** The McGraw-Hill Companies, Inc, USA

29. ReVelle J. (2004), ***Quality Essentials: A Reference Guide from A to Z***, ASQ Quality Press, 152–155.

30. Sangwan R., Bass M., Mullick N., Paulish D., and Kazmeier J (2007), ***Global Software Development Handbook***, Taylor & Francis Group, LLC, USA

31. Shrivastava S. and Date H. (2010), ***Distributed Agile Software Development: A Review***, Journal of Computer Science and Engineering, Vol.1, 10-17.

32. Sommerville Ian (2011), ***Software Engineering, 9th edition***, Pearson Education Inc., USA.

33. Sone S. (2008), ***Mapping agile project management practices to project management challenges for software development***, Doctoral thesis ,Argosy University

34. Tenhunen T. (2010), ***Challenging in Scaling Agile Software Development***, Tampere University of Technology, Master Thesis.

35. Ylimannela, V. (2010), ***A Model for Risk Management in Agile Software Development,*** [Accessed: 13 Sep, 2013] http://www.cloudsw.org/under-review/1af0cbb6-6d4c-4082-8530-17e1317e2ceb/a6f468c9-4857-4206-96ee-f67df0583d41