UPPSALA
UNIVERSITET

Exploring the difference between Agile and Lean:
A stakeholder perspective

Mohammad Shahidul Islam

Sentayehu Tura

Uppsala University
Department of Informatics and Media

UPPSALA
UNIVERSITET

Exploring the difference between Agile and Lean:
A stakeholder perspective

Mohammad Shahidul Islam

Sentayehu Tura

Supervised by:
Pär Ågerfalk

Thesis for the Degree of Master of Science in Information Systems
Department of Informatics and Media,
Uppsala University,
UPPSALA, Sweden
02 May, 2013

Abstract

In this thesis, we have identified the difference between Agile and Lean methods based on stakeholder's perspectives. To achieve the goal we have deal with only Agile and Lean principles. In addition, in order to identify the stakeholders from Agile and Lean principles we have used the relevant practices from both sides. As the principles of Agile manifesto are directly followed by most of the organizations, we have also used Agile principles directly in this research. On the other hand lean methods have no pure principles, as a result we have used the most common and popular lean principles derived from different authors. We have only considered the most relevant principles that might be useful in software development. To achieve a stronger result of this thesis we have also considered stakeholder theory. Moreover we have identified the stakeholder's involvement with Agile/Lean principles and stakeholder theory.

Key words: Agile methods, Lean thinking, Lean software development, Stakeholders, Stakeholder theory.

Acknowledgement

This thesis came to completion after a long journey of encouraging and constructive feedback given to us by our adviser, Professor Pär J. Ågerfalk, whom we would like to express our heartfelt gratitude. We also would like to thank senior lecturer Jonas Sjöström for encouraging us to work on this topic in the first place. And last, but not least, to family and friends, thank you all for your patience and wonderful support.

Contents

Abstract 3

Acknowledgement 4

Contents 5
List of tables: 7

1. Introduction 9
1.1 Motivation 10
1.2 Research questions 11
1.3 Thesis limitation 11
1.4 Structure 12

2. Research Method & Analytical Framework 13

3. Stakeholder Theory 17
3.1 Introduction 17
3.2 Stakeholder Theory Models 18
3.3 Different Models for Identifying Stakeholders 21

4. Agile methods 24
4.1 Introduction 24
4.2 Agile principles 25
4.3 Practices and Tools 30
4.3.1 Scrum 30
4.3.2 Extreme Programming 33
4.4 Summary and Definition 38

5. Lean methods 40
5.1 Introduction 40
5.2 Lean principles 42
5.3 Practices and Tools 50
5.3.1 Just-in-time 50
5.3.2 Kaizen 50
5.3.3 Queuing Theory 51
5.3.4 Leadership 51
5.3.5 Cost of delay 52
5.3.6 Value stream mapping 52
5.4 Summary and Definition 53

6. Derived Agile and Lean principles 55
6.1 Agile principles based on stakeholder perspective 55
6.1.1 Working software frequently 55
6.1.2 Embrace change 56

6.1.3 Simplicity....58
6.1.4 Customer satisfaction....59
6.1.5 Stakeholder involvement....60
6.1.6 Motivated and self-organizing teams....61
6.1.7 Sustainable development....63
6.1.8 Technical excellence....64
6.1.9 Face-to-face communication....65
6.1.10 Product improvement....66
6.2 Lean principles based on stakeholder perspective....67
6.2.1 Define customer value by eliminating waste....67
6.2.2 Decide as late as possible....69
6.2.3 Move towards flow and deliver values fast....71
6.2.4 Continuous improvement....72
6.2.5 Create learning environment....73
6.2.6 People focus....75
6.2.7 Customer focus....76
6.2.8 Product excellence....77
6.3 Summary....81

7. Analysis....82
7.1 Comparison work....82
7.1.1 Agile principles Vs Lean principles....82
7.1.2 Stakeholder theory Vs Agile and Lean stakeholders....89
7.2 Differentiating stakeholders based on Mitchell's model....94
7.3 Summary....98

8. Findings and Conclusion....99
8.1 Future work....101

9. References....102

Appendix: Glossary of Terms....108

List of figures:

Figure 1: Jarvinen's taxonomy of research methods (Research Questions Guiding Selection of an Appropriate Research Method)....14
Figure 2: Four tier model for identifying relationship between stakeholders and principles.....16
Figure 3: Traditional stakeholder theory....19
Figure 4: Basic two-tier stakeholder map (source: Freeman et al. 2008:51) 20
Figure 5: Basic two- stakeholder map (source: Freeman et al. 2008:62)....21
Figure 6: Stakeholder typology (Mitchell, Agle and Wood, 1997:874) 22

Figure 7: The Scrum process 31
Figure 8: Partitioning of Agile project stakeholders by (Koch, 2005).....61
Figure 9: Possible nearest Agile versus Lean principles.....83
Figure 10: Stakeholder theory relationship with derived Agile principles ...91
Figure 11: Stakeholder theory relationship with derived Lean principles ...93

List of tables:

Table 1: Lean principles mentioned by different authors 48
Table 2: Derived lean principles based on different authors.....80
Table 3: Identifying stakeholders based on Mitchell's model 96

Abbreviations

ASD Agile Software Development
BFS Breadth-First Search
DFS Depth-First Search
DSDM Dynamic Systems Development Methodology
FDD Feature Driven Development
GSD Global Software Development
JIT Just In Time
LSD Lean Software Development
TPS Toyota Production System
XP Extreme Programming

1. Introduction

Today's fast growing technology, diverse business challenges and demanding customer requirements force organizations to seek and discover strategies or methods that can handle such problems. Lean and Agile methods are amongst the different methods that are capable of solving these kinds of problems. At the beginning Lean is about manufacturing and production but its high level goals and principles can also be applied to software development. Whereas Agile methods did not appear out of thin air but were based on different practices from earlier times. It is well known by its values of customer collaboration, iterative development, self-organizing teams, and adaptability to change (Rico, 2010).

The aim of agile software development is to overcome the limitations of plan-driven software development like welcoming change of requirements at any time. It focuses on close cooperation between customers and developers, delivering software within time and budgets constraints. Agile development process is repetitive, adaptive and minimally defined as it relays on frequent informal face-to-face communication. There are different types of Agile methodologies where extreme programming (XP) and Scrum are widely used (Jalali and Wohlin, 2010). The advantage of using agile methodology is to get benefits like flexible project management, minimize product cost, increase communication and finally that can lead to customer's satisfaction. On the other hand, lean is not too far away from software development. Different authors advocate that the idea behind lean manufacturing might be useful for software development (Poppendieck, 2003; Middleton, 2001; Middleton, Flaxel &Cookson, 2005; Poppendieck & Cusumano, 2012). "*In 2008/09, BBC Worldwide generated profits of £103 million on revenues of £1.004 billion*" (Middleton and Joyce, 2012 cited in BBC Worldwide website, 2010, p. 21) using lean thinking in software development.

However, recently different authors suggested combining agile and lean together for better software development, although authors argue on the difference between agile and lean. Difference between agile and lean could be considered from two views. One view is that both have their own different levels, like lean could be considered as a set of principles where as agile is more practical. Another view of difference is from their scope and focus, where Agile methods concern on specific product development only. On the

other hand lean concerns on everywhere like business context where developing product is a small part of it (Wang, 2011). Naylor, Naim and Berry(1999) defined "*agility means using market knowledge and a virtual corporation to exploit profitable opportunities in a volatile market place where leanness means developing a value stream to eliminate all waste, including time and to ensure a level schedule*".

Stakeholder theory comes from the area of strategic management where Freeman and Reed (1983) defined stakeholder is "any particular group or individual who can affect the achievement of organization's objective or is affected by the achievement of organization's objective". The concept of stakeholder theory in software development is always ignored although it is well recognized in the field of IS. The principles and concept of stakeholder theory may apply in software development that can help managers to understand roles and responsibilities from different stakeholder's point of view that may also influence product development efforts. The most two common scenarios always observed at the end of the software development are that either customers reject the product as it does not meet their requirement or the cost and schedule of the project become too large. To avoid such situations McManus (2004) suggested collecting three types of information about stakeholders: information about each stakeholder separately with their roles and profile, information about how to interact with them and finally about their interaction with other stakeholders.

In this research we will illustrate the difference between agile and lean based on stakeholder's perspective. To achieve this goal we use the principles from both agile and lean. We also identified relevant stakeholders for each agile/lean principle through practices which will help managers to choose proper practice for specific principle concerning relevant stakeholders. And finally we will use Mitchell's model to differentiate several stakeholders for each principles.

1.1 Motivation

There are two motivations for this thesis work. First motivation is why managers should combine agile/lean methods and the second motivation is why managers need to consider stakeholder theory.

Polk (2011) who was the team manager of NG (Networking Gaming) system team development of WMS Gaming Inc tried to get the benefits using agile software development. He tried to run his project based on agile principles. He divided eighteen members in three teams and later on worked with a

single team. In both cases he failed rather he identified the team was not self organized. He found three to five members were failing to work independently. Later on, he introduced Kanban method on his team where the whole team divided in two teams known as Iterative team and Kanban team. Iterative team was responsible for developing large scale project based work where as Kanban team was responsible for developing smaller features. Also Kanban team was primarily consists of those members who were unable to work independently or performance below the expectation, also failing to complete work inside iterations. He also arranged training for Kanban team and finally he succeeded to run his project.

The advantage of involving stakeholders in every product development is obvious where as ignoring stakeholders is one of the key reasons of the project failure. One of the main reasons behind the project failure is lack of user involvement where statistics show that 34% of projects succeed in 2004, 35% in 2006, and 32% in 2009. Even report shows that a company had to lose $20 million because of ignoring stakeholders. Moreover, 86% developers are agreed to concern on stakeholder analysis because of its importance (Lim, Quercia & Finkelstein, 2010).

1.2 Research questions

There are two main research questions:

1. What are the differences between agile and lean methods based on a stakeholder perspective?
2. How can managers benefit from combining agile and lean methods?

1.3 Thesis limitation

In this research overall Agile/Lean principles were based on literature review. Even our whole identified stakeholders were derived from literature. For Agile section we used directly Agile alliance principles as it is mostly followed by most of the software development companies. On the other hand Lean method is new in software development areas where there are no pure Lean principles. We have gathered Lean principles from different well-known authors and merge them only that are relevant to software development. There were no empirical data in our research, although we believe that it was possible to make this research result stronger while considering empirical data.

1.4 Structure

The remaining part of this thesis is as follows: Chapter 2 contains methodology part, how we come up with a common list of principles from agile and lean methods. Chapter 3 consists of stakeholder theory with different models. Chapter 4 consists of agile principles derived from agile manifesto and then agile methods including different practices (XP, Scrum). Chapter 5 consists of lean principles suggested by different authors which also followed by lean methods including different practices. Chapter 6 describes briefly agile and lean principles based on stakeholder perspectives. Chapter 7 is the heart of our thesis which includes two ways of comparison work. One ways is agile principles versus lean principles and another ways is stakeholder theory versus agile and lean principles. Chapter 7 also includes Mitchell's model where it shows the importance level of different stakeholders. And finally chapter 8 includes a detail conclusion with findings and future work of our thesis.

2. Research Method & Analytical Framework

This paper mainly used the principles and practices of ASD and LSD methodologies in order to identify the underlying difference between ASD and LSD in relation to different stakeholders. Senge (1990) described practices as activities where as principles as "guiding ideas and insights" to the activities. There are different principles and practices in ASD and LSD but we have considered using only agile manifesto principles for ASD and for LSD we concentrate on different scholars derived lean principles (principles those are only related to software development). However for the analysis of Lean principles sources that are considered as the leading contribution in Lean thinking are given priorities. The likes of Ohno, Womack and Jones (2003), Poppendieck, Charette etc provide useful insight for finalizing the principles of our choice. Not to forget the fruit full works of Middleton and other's papers and articles about the use of Lean thinking in software development. For the information about stakeholder theory, we have used the books, and papers from the people who most are known for it, people like Freeman, Harrison, Wicks, Parmar, Mitchell, Agel & Wood, Friedman and so on.

A systematic approach has been used to carry out the search and selection for the literature. Different searching mechanisms have been conducted including keyword searches for books, relevant papers and websites using web of knowledge and by going through several specific highly rated academic journals. Initially, first read of the materials were performed to identify the proper concept of what they are about. For published articles, a look at the abstracts were taken first in order to decide if it's worthy of further reading or inclusion to reference list. After we decided which ones to use in second selection, we did an initial classification or grouping of our articles by type of sources. The next step was to go through each material in order to carry out further systematic and critical review of the content. While doing that it was necessary for us to follow a method called the preview, question, read, summarize (PQRS) system. It helps to be focused, consistence and also for easy identification and retrieval of information since we are assessing a large number of materials. Moreover, we have used Meta-synthesis technique for our analysis and findings sections.

The above sources' analysis brought in several sets of principles, especially for lean. In order to come up with the appropriate ones within the context of our paper, we did the analysis in iterative fashion. While doing that, the iterative synthesis involved grouping of the principles from narrow to broad and also from complex to specific. In addition a clear description of each principle is given for understanding the meaning of the principle in order to pin point the difference underlying between these two methodologies. To do that first we have compared principle versus principle from both sides and second we performed an analysis of stakeholder theory versus Agile and Lean principles separately.

Based on our research question we believed it's best to deal with it analytically using Max Weber's notion of ideal types rather than dealing with it empirically. For this thesis we have use Conceptual-analytical research approach as a research methodology. In Conceptual-analytical research, "the basic assumption underlying constructs first analyzed; theories, models and frameworks used in previous empirical studies, are identified, and logical reasoning is there after applied" (Järvinen, 2000).

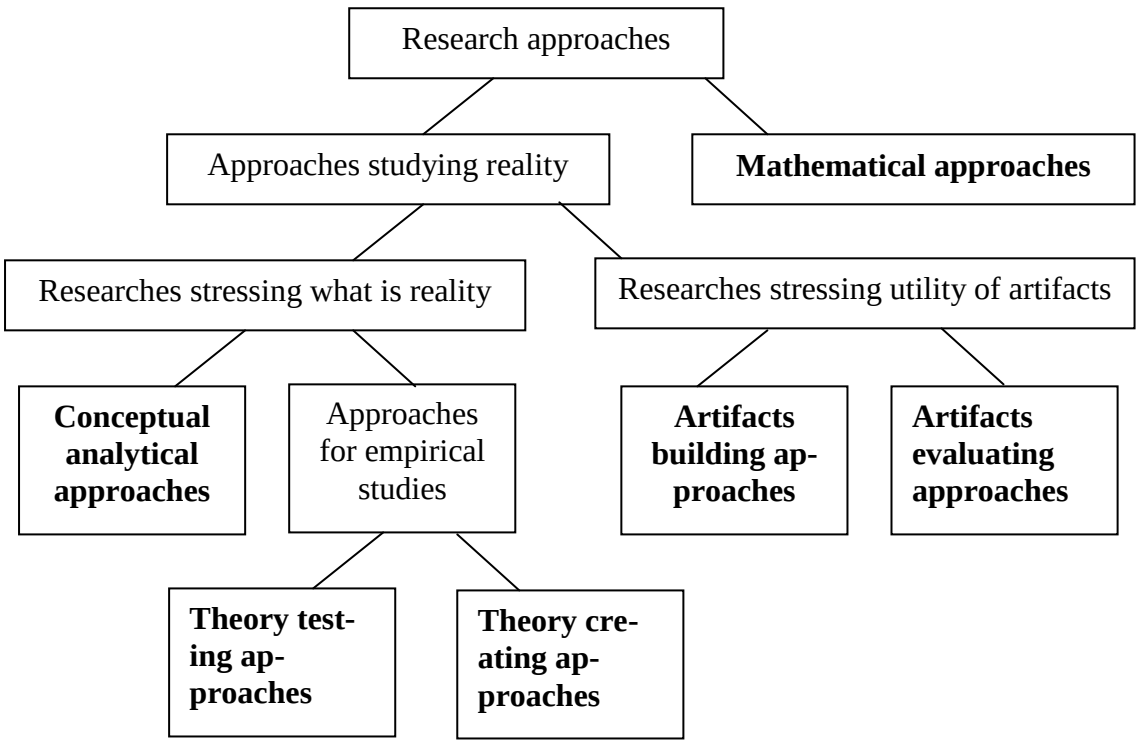

Figure 1: Jarvinen's taxonomy of research methods (Research Questions Guiding Selection of an Appropriate Research Method)

We have chosen a conceptual-analytical approach (Järvinen) based on Weber's notion of ideal types. An ideal type is formed from characteristics and elements of the given phenomena, it doesn't refer to perfect things, moral ideas or statistical averages but rather stress on certain elements common to most cases of the given phenomena. According to Weber, an ideal type is a useful tool for comparing and analyzing social or economic phenomena. We reconstruct the ideas, based on established well-cited literature on two ideal types "agile" and "lean" in order to compare conceptually-analytically. One thing we should bear in mind is that, ideal types do not exist as such in real life but studying ideal types can help us understand complex social phenomena in a way that facilitate empirical studies subsequently.

Koch (2005) writes on the book called "Agile software development: Evaluating the methods of your organization", people processes and tools are all important for the success of software projects. The author uses an example of putting the project on three legged table and each leg represents people, process and tools, and if one leg is broken the table could collapse. So in order to keep the balance three of them need each other's company or in other words, they need each other in order to succeed. According to Liker and Morgan (2006, pp. 9) people are "who work hard as a team to achieve common objectives". On the other hand, roles and responsibilities of the project, action taken by people and developed products are identified by processes. "The process starts with specific stretch objectives for each program and the teams virtually always achieve the targets" Liker and Morgan (2006, pp. 9). Tools or technologies may considered as a helping objects that may improve the process where Liker and Morgan (2006, pp. 9) defined "Technology to Toyota is a set of tools to enable the people to execute and improve the process- no more and no less". A very well known quote by one of the Toyota Vice President about tools is that "Computer technology does not change the way we work. It simply help us do what we do faster", Liker and Morgan (2006, pp. 9). Using this philosophy we have come up with an analytical frame work that classifies the different principles and practices of ASD and LSD under the category of people, process and tools. Because some principles tend to indicate to peoples, some presumes to be more like processes and others sound more like tools. However there are believes and wrong interpretations, for example in ASD it is considered that people have more importance than processes and tools, whereas LSD is assumed to give more importance to processes and tools than people. Therefore by dividing the principles from both sides in to people, process and tools we will show which are the practices goes under each category with identifying what type of stakeholder involved with it. The figure below shows this classification.

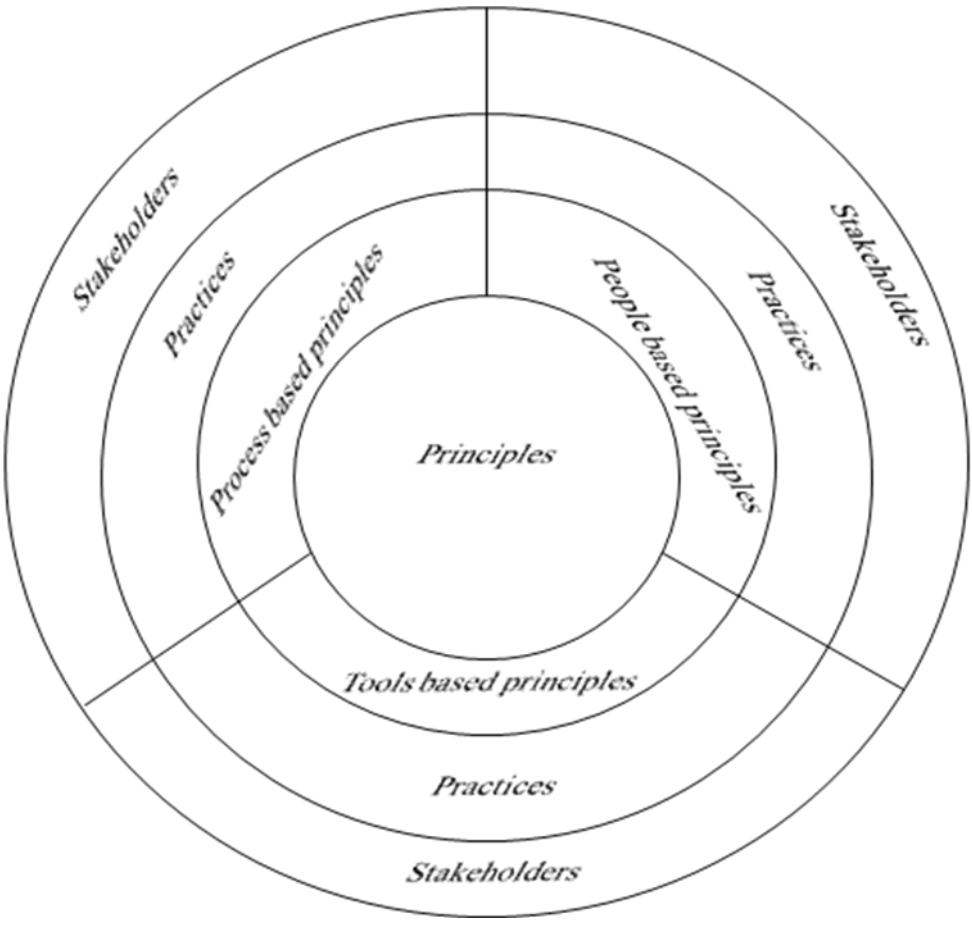

Figure 2: Four tier model for identifying relationship between stakeholders and principles.

From the above diagram it is clear that to identify the stakeholders either in Agile or Lean areas, first principles have been considered at the beginning. As there are huge numbers of principles especially in Lean section therefore, principles have been divided under process, people and tools (Liker and Morgan, 2006) i.e. few principles goes under process based principle, few principles goes under people based principle and so on. Again from each principle group either process based principle or people based principle or tools based principles has its own practices through which is it is a good idea to identify stakeholders. Of course there is no room to become confused about People based principles and Stakeholders in this research. People based principles means only those principles that are directly related with people or person (for example motivate employees, empower the team etc) where stakeholder itself is a group of people.

3. Stakeholder Theory

3.1 Introduction

Several stakeholder definitions are proposed from different scholars from diverse disciplines with different points of view. As cited in the book of Freeman (1984), the word “stakeholder” originally discovered in the management literature by the Stanford Research Institute (SRI) in 1963 and it is defined as "those groups without whose support the organization would cease to exist." At that time the possible stakeholders identified were shareowners, employees, customers, suppliers, lenders and society. For the survival of the firm, the need and concerns of these groups of stakeholders should be understood by the executives, note the SRI researchers Freeman (1984). However Freeman and David (1983) argue how general this definition is and they proposed two broader definitions of stakeholder: a wide sense and a narrow sense. In wide sense it is any particular “group or individual who can affect the achievement of organization’s objective or is affected by the achievement of organization’s objective.” Under this category, they included different group of stakeholders such as: Public interest groups, protest groups, government agencies, trade associations, competitors, unions, as well as employees, customer segments, shareowners, and so on. In the other hand, for narrow sense of stakeholder, organization has to depend on a particular group or individual for its survival. The stakeholders could be Employees, customer segments, certain suppliers, key government agencies, shareowners, certain financial institutions, and so on.

The development of stakeholder theory goes deep in history to trace with enormous credit given to the contribution of diverse theorists from a variety of disciplines. It is considered as a framework for other theories. “Stakeholder theory is fundamentally a theory about how business works at its best, and how it could work. It is descriptive, prescriptive and instrumental at the same time.” Freeman et al (2010). It is constructed to solve the problems of “value creation and trade”, “ethics and capitalism”, and “managerial mind set.” To solve these three problems, the stakeholder theory suggests developing a relationship between the business and the different stakeholders who have a stake on the business.

Donaldson & Preston (1995) identified three categories of stakeholder theory. Descriptive/Empirical stakeholder theory is used to describe some corporate characteristics such as the nature of the firm, how managers think about managing, the way board members think about the interest of corporate constituencies and how corporations are managed. The second type of stakeholder theory is Instrumental where it's used to make sure whether there is a connection or not between stakeholder management and the achievement of traditional corporate objectives. Normative is the third type which has to do with identification of moral or philosophical guidelines for management and functioning of corporations.

Managers can manage the stakeholders best if they know who brings the most value to the business. Freeman, Harrison and Wicks (2008), discuss about categorizing the different groups of stakeholders in to primary stakeholders (customers, suppliers, financiers, communities and employees) and secondary stakeholders (media, special interest groups, consumer advocate groups, competitors, and government), depending on the impact they have towards the firm. Although the degree of importance and stake differs, managers, customers, suppliers, employees, financiers, and communities play an important role for the achievement of today's business organization. Keeping the interest of this different group of stakeholders could be a difficult job. But still Building and leading a great company has always been about managing for stakeholders. After all, stakeholders are the building blocks of a competitive firm.

3.2 Stakeholder Theory Models

Since the development of the traditional firm-centric stakeholder map by Freeman, a lot has changed in stakeholder concept, including the definition and the development of different stakeholder model. Different scholars try to provide a variety of stakeholder models with diverse point of view. For example Freeman provided firm-centric stakeholder models where as Werhane provided decentering stakeholder model. In 1988 Freeman provided a stakeholder model where the firm is put in the center of the graph (Phillips, 2011). On this type of model the firm or its managers are the primary targets and the stakeholders are considered in a second basis. The arrows show accountability relationships among the different group of stakeholders.

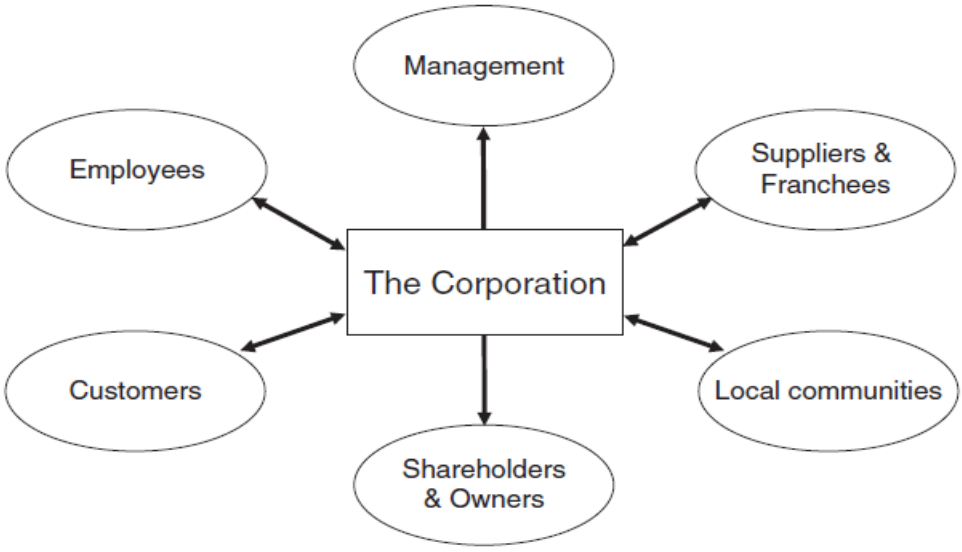

Figure 3: Traditional stakeholder theory

Later on Freeman, Harrison and Wicks (2008) published a book "*Managing for stakeholders: survival, reputation and success*" described a basic two tier model keeping the firm in the center for identifying stakeholders. In the first tier customers, employees, suppliers, communities and financiers have been kept because they are essential to continued growth and survival of any business and known as primary stakeholders or definitional stakeholders. In the other hand, government, competitors, media, environmentalist, corporate critics, special-interest groups are kept in the second tier known as secondary stakeholders. Moreover specific company's stakeholder maps may differ from figure 1.1. For example if the company works for defense industry then the government will become primary stakeholder.

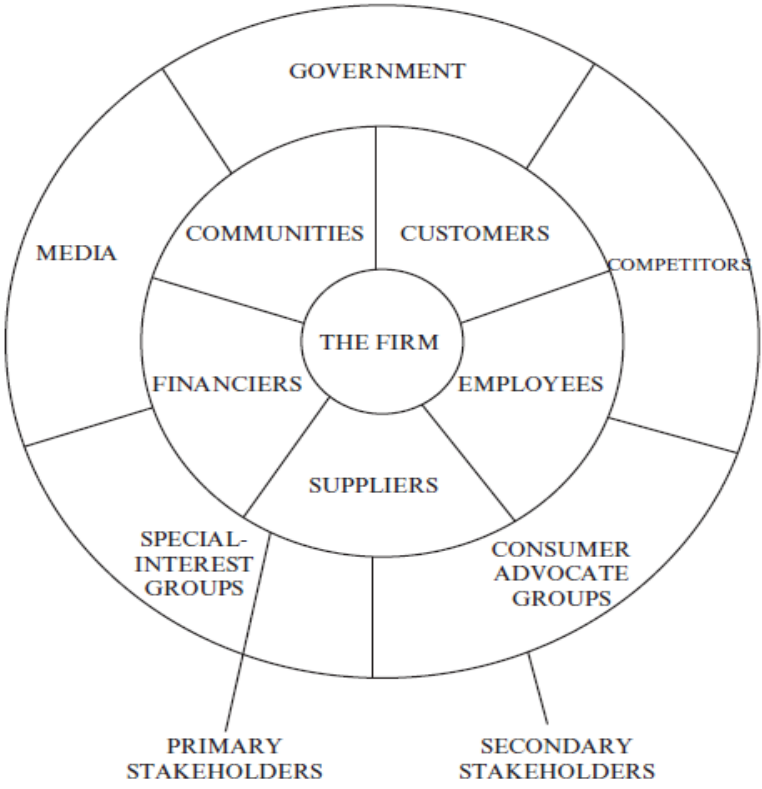

Figure 4: Basic two-tier stakeholder map (source: Freeman et al. 2008:51)

Freeman later on modifies his firm-centric model by putting the "manager in a job" at the center of the circle in a connection with all the stakeholders considered as internal stakeholders and external stakeholders. On this model the manger has a better way of reaching to all different groups of stakeholders which in the way helps the manager to fulfill his or her responsibilities better. However, the model puts the manager in the center, who works for the firm so it's still firm-centric model.

Werhane on (Phillips, 2011) discusses about the impact of decentering stakeholder models. She suggests four ways to deconstruct a firm-centric stakeholder model which can affect our thinking towards corporate responsibility. One way could be by putting a stakeholder in the middle in place of the firm; however this stakeholder should not be part of the firm. The second mechanism to bring attention could be by placing a real picture of individual stakeholder in the center of the graph, which can be an employee. Thirdly the author argues taking a systems approach to stakeholder i.e. using the interconnection between people's practices and different institutions. The

fourth type of decentring method is by creating partnership with global organizations.

3.3 Different Models for Identifying Stakeholders

Freeman, Harrison and Wicks (2008) provide ten principles on managing stakeholders. While applying these principles three level of thinking are suggested, where one of them is "identify who are the critical stakeholders for each business?" But still identifying stakeholder is unclear. Of course there must be more clarification on each stakeholder. Later on the following complex stakeholder model has been provided.

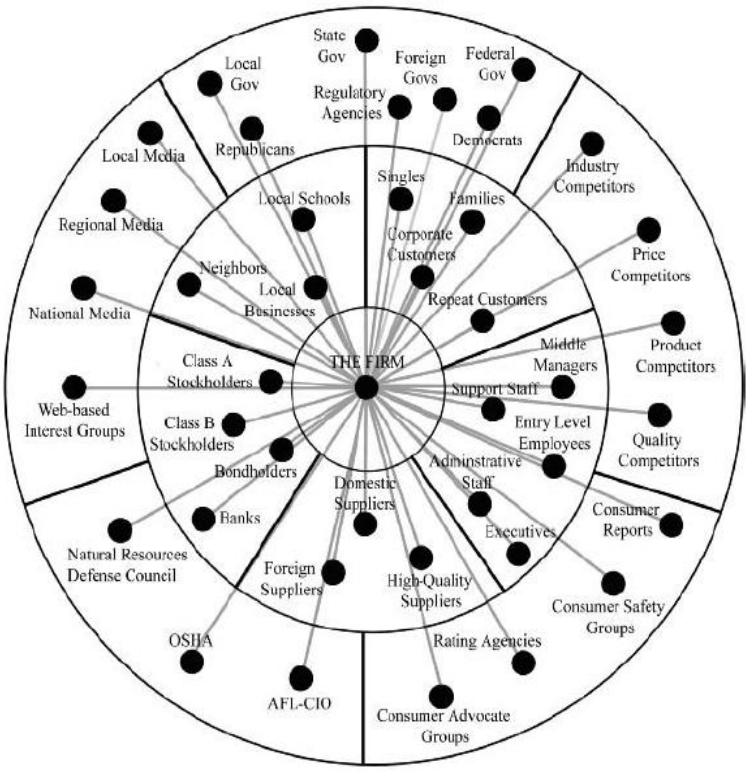

Figure 5: Basic two- stakeholder map (source: Freeman et al. 2008:62)

Stakeholders need to be identified at the generic level, as shown in figure 1.2. In addition, they need to be identified at a finer level of analysis. This can be done by segmenting the stakeholders in to more meaningful catego-

ries. For instance, instead of taking customers in general as a key stakeholder, categorizing it into different classes would be more efficient and helpful.

Identifying who and what really matters inquire managers to pay attention to different classes of stakeholders. The various classes of stakeholders possess different attribute towards the firm. Mitchell et al (1997) came up with a model that suggests the stakeholder's impact or perception to organization can be categorized by the manager into *power, legitimacy* and *urgency*. The three circles, on Figure 1.4, presented separately as power, legitimacy and urgency. Each circle may be part of overlapping with one another to be further classified into seven levels of perceived values of the stakeholders by managers. Each type of the seven stakeholders may possess all the three attributes or two of the attributes or one of the attribute. However, according to the model, managers will not consider entities with no power, legitimacy, or urgency in relation to the firm as stakeholders. In addition the managers will not consider these entities to have a salience.

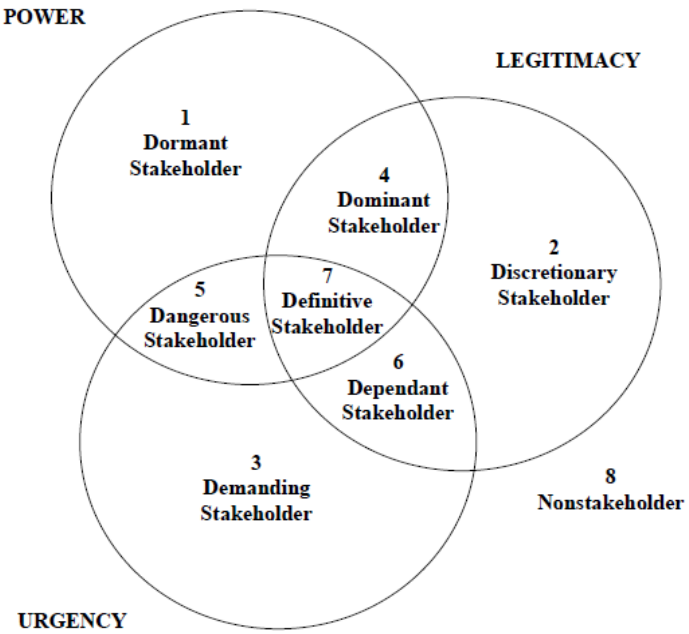

Figure 6: Stakeholder typology (Mitchell, Agle and Wood, 1997:874)

As shown in the figure, if stakeholders possess only one of the three attributes, and include the behaviors of dormant, discretionary, and demanding stakeholders, then they are called *latent stakeholders*. Latent stakeholders only posses one of the defining attributes as a result managers may not give much attention to them and also these types of stakeholders don't probably

give attention to the firm either. Stakeholders with two of the attributes, and include dominant, dependent, and dangerous stakeholders, they are labeled as *expectant stakeholders*. Expectant stakeholders posses two of the stakeholders' attributes as a result one can expect a strong relationship between them and the manger. Stakeholders possessing all the attributes (power, legitimacy, and urgency) are called *definitive stakeholders*. These type of stakeholder's posses all the three attributes which makes them very important stakeholders and stakeholder salience would be higher. Finally non stakeholders don't posses any of the three attributes.

There is no specific rule on how to find stakeholders or who to consider as one or who to involve. One method may work best for one and fails badly for another. According to the World Bank (1996) report, the best way to identify appropriate stakeholders could be by asking questions such as:

- Who might be affected (positively or negatively) by the development concern to be addressed?
- Who are the "voiceless" for whom special efforts may have to be made?
- Who are the representatives of those likely to be affected?
- Who is responsible for what is intended?
- Who is likely to mobilize for or against what is intended?
- Who can make what is intended more effective through their participation or less effective by their non-participation or outright opposition?
- Who can contribute financial and technical resources?
- Whose behavior has to change for the effort to succeed?

After identifying the appropriate stakeholders, the next task could be finding a way to create relationship so that they can be involved. Continuous interaction and sharing information with stakeholders can be key factors for building trust. Stakeholder theory advocates, after identifying the stakeholders we need to assess what kind of effects (political, economic, social, or managerial) each stakeholder have on the firm and what value they add to the firm (Freeman, 1984).

4. Agile methods

4.1 Introduction

Agile methods, Agile software development, Agile project management, Agile manufacturing, called in different ways but all refer to a methodology that did not appear suddenly. Even though the methods are well known in 1990s, the principles existed before 1975. Agile methods were firmly based on "autonomous work groups from the 1950s", "end user involvement from the 1960s", iterative and incremental development from the 1970s, and joint application design methods and rapid application development from the 1980s (Rico et al, 2008). Agile methods constitute various significant advantages over some traditional software developments; one perceptible advantage could be, in traditional software developments, there are bouquets of unnecessary documentations to deal with, that are resulted from change of requirements throughout the project's lifetime. Moreover, according to Perera & Fernando's (2007) point of view, Agile methods provide "project management, cost effective adaptability, increase communication and ultimately increased customer satisfaction". Furthermore; Beyer (2010) argues how Agile methods pioneer a new way of developing software to the field by declaring the values and attitudes the team should possess. This means, encouraging the team to avoid the habit of designing before coding, avoiding unnecessary documentation, and reminding the team face-to-face communication and collaboration are key factors for successful development.

One may answer what agility really means before going to discuss the different principles and practices agile methods may possess. According to Highsmith, 2002, "agility is the ability to both create and respond to change in order to profit in a turbulent business environment." Agile organizations embrace change, create change, and handle change better than their competitive advantages and do not consider change as extra cost. However, companies must identify the level of agility they need to incorporate to maintain their competitiveness. "A copper mining company doesn't need to be as agile as a biotechnology firm" (Highsmith, 2002). There are different types of agile methods that are emerged and popular include; Crystal Methods, Scrum, Dynamic Systems Development Methodology (DSDM), Feature-Driven Development (FDD), and finally Extreme Programming (XP). All

these methods share common practices that make them Agile: short, well-defined iterations, minimal documentation, tight team processes, and continual feedback from stakeholders (Beyer, 2010).

Agile project management is guided by a manifesto and sets of principles that are discovered in 2001 by a group called the Agile Alliance (www.agilealiance.com). The group came up with the so called the agile manifesto, which is most of the agile practices are guided by these more general values but the principles are more detail. The agile manifesto is based on four broad values: "*Individuals and interactions* over processes and tools", "*Working software* over comprehensive documentation", "*Customer collaboration* over contract negotiation", and "*Responding to change* over following a plan". The Manifesto agrees that there are values on the items on the right but they value more for the items on the left, unlike the traditional software development methods (Cockburn, 2002). Agile methods are created so that the problems of both developers and management could be solved. Agile methods can give the satisfaction of being powerful and in control for developers. At the same time, management won't be kept in darkness waiting and wondering what the project would be like. Agile methods incorporate principles that provide a mechanism to control the chaos of software projects.

4.2 Agile principles

Larman (2004) argues how it is difficult to define what Agile methods are, since there are various types of basic practices, but there are practices that are shared by different methods such as; short time boxed, iterative development, adaptive planning, evolutionary delivery and so on. Beyer (2010) also seems to agree how the core agile methods share the same principle. Which in his book, User-Centered Agile Methods mentioned, as; "short, well-defined iterations that deliver real user value; tight team processes for efficient development; minimal documentation of specifications; and continual feedback from stakeholders to validate progress".

On the book, "Agile Principles Unleashed: Proven approaches for achieving real productivity gains in any organization", Cooke (2010) discuses the 12 core principles in which the success of Agile approaches depended on. The author writes how these principles combined can create a business environment that produce "high business-value output", "motivates employees", "encourages innovation" and "delivers tangible results". The 12 Agile principles, according to Cooke (2010), are:

- Responsive planning: this has to do with breaking down the whole system into shorter iterative delivery cycles and depending on the outcome of the delivery cycle, one can adapt and continue.
- Business-value-driven work: involves prioritizing the organization's work according to primary and secondary business value outcomes
- Hands-on business outputs: making sure the business requirements are met and business values are delivered for the organization by checking outputs often.
- Direct stakeholder engagement: active participation of both the internal and external customers throughout the process in order to fulfill their expectations.
- Immovable deadlines: Team members are more committed and have the urgency to work towards fixed deadlines than flexible deadlines; it helps the organization get ongoing business value.
- Management by self-motivation: trusting self motivated and organized employees do the job in their own way rather than telling them how to do it.
- 'Just-in-time' communication: changing the traditional corporate meetings to regular and active face to face communication for better knowledge transfer and effective learning.
- Immediate status tracking: an ongoing activity which is expected from the delivery team members to keep everyone in the organization, including business owners, aware of the progress of the work they are doing.
- Waste management: eliminating anything in the organization that cannot lead to high business-value outputs, anything which wastes the time, money and resources of the organization.
- Constantly measurable quality: ''involves creating active checkpoints where organizations can assess outputs against both qualitative and quantitative measurements''
- Rear-view mirror checking: provides different tools, such as retrospectives, for the staff, ''for regularly monitoring and self-correcting their work''.
- Continuous improvement: to make sure organizations are keeping the pace with the current technology, maintaining a competitive advantage and fulfilling the customers' demand, the work being done is in continuous improvement in each iteration with the involvement of the stakeholders.

People have different way of describing the different Agile principles in order to fit their need and desire. However for this paper, pure Agile Manifesto principles are used as they are, unpolished. Agile principles from the Agile alliance (Cockburn, 2002):

1. Our highest priority is to satisfy the customer through early and frequent delivery of valuable software: Agile development is focused on delivering software continuously to the customer so that there can be greater advantage on getting early feedback, it can be about the requirement, the team and the different software processes. The time for delivery should be given great consideration and negotiation with the customer so that the changes that are made will not disturb the user. Instead of delivering a software that may take a year or longer, this principle emphasizes frequent delivery of valuable software for the customer and even if the project can't go on but still the customer has a working software (Cockburn, 2002).

2. "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale": The statement emphasizes the length of the iteration or working cycle for the team to deliver a working software should be faster depending on the user's acceptance or reaction to the changes. Cockburn (2002) writes: "If the users can accept changes every month, and the development team can match the ongoing requests for changes, then the shorter feedback cycle is better." This principle is commonly known as "continuous delivery", which ensures that the project is going on the right way and this gives greater satisfaction for customers. Koch (2005) suggested 2-8 weeks are best for each incremental interval.

3. "Working software is the primary measure of progress": This principle gives emphasis to delivering a running code rather than "promissory notes in the form of plans and documents". No Matter how large the project is, Agile methods advice to break down features into smaller pieces, so that it can be incrementally implemented and tested (Cockburn, 2002).

4. "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage": "If agile methods have a motto, it is embrace change. If agile methods have a strategic point, it is maneuverability" writes Larman (2004). Some of the processes of Agile development such as: early and frequent delivery of running software, time-boxed iterative and evolutionary development, continual attention to architecture, and motivation to update the design trigger agile methods into willingly accepting changing requirements even late in the development process. "If your company can deliver quickly and respond to late-breaking information and your competitors company can't, then your company can out-maneuver your competitors on the software front" Cockburn (2002) explains about the maneuverability behavior of Agile methods. Agile methods are designed in such a way that it can adopt any challenging situation. As changing requirement is a part of project so developers should

welcome this situation often. Though such situation may become challenging sometimes but it is the key to survive in exigent market (Koch 2005).

5. "Business people and developers work together daily throughout the project": The principle emphasizes onsite participation and discussion of business experts and developers can only benefit the success of the project. The developers will get new information rapidly through communication with the business people and this on the way helps to make early decisions and the sponsors can make sure that they got what they needed (Cockburn, 2002).

This principle shows the partitioning of the project stakeholders while developing software; it divides two groups of stakeholders, developers and business people. Developers are the members of technical team who are responsible from technical side like programmers, system architects, testers, technical writers etc. on the other hand business peoples involve from management structure (senior and middle management and sometimes project manager), supporting services (human resources, finance, contracts and information technology), customers and end user. In Agile "business people" refers everyone who are involved with the project. Koch (2005) suggests better way to state this principle as "All of the project stakeholders must work together with the development team daily throughout the project".

6. "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done": Agile methods underline the need of having motivated and skilled individuals self organizing and communicating by their own and making things happen than forcing people to work and follow some kind of well defined process. The principle outlines, to create the working environment and the facilities needed but to let people make their own decision (Cockburn, 2002).
Agile methods believe that team members will perform their work on their own judgment rather than simply following orders for what they are assigned. Sometimes it will not be possible to solve problem by them selves, then Agile methods expect team members may raise questions to discuss with the team, management and customer. Agile methods not only believe on motivated individuals and motivated team, moreover it ensures appropriate environment, professional support, and fully trust to let them work on their judgment. Koch (2005) suggested a better way to state this principle: "Build project environments that generate motivated individuals".

7. "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation"; Agile methods suggest face-to-face communication rather than any types of other form of communication (specially written type documentations), Koch (2005).

Face-to-face communication is the most efficient way of communication rather than any other communication channels. Statistic shows that 79% communication occurs via face-to-face, 17% from email and only 4% from telephone. Since there is a possibility of losing information in face-to-face communication, different tools like papers, white-boards can be used for storing information.

8. "The best architectures, requirements and designs emerge from self-organizing teams"; the architecture of the system should progress with "the changing knowledge of a team and the changing wishes of the user community" (Cockburn, 2002). Agile philosophy is far away from traditional command and control management where it encourages self-managed, self-directed or self-organized team. Agile team is an entity where it has its own knowledge, perspective, motivation and expertise. In this sense team can be consider as a partner with its management and customer where it has its own decisions and negotiation commitments (Koch, 2005). A brief explanation of self-organize team from practical experience is described in Hoda et al (2011).

9. "Continuous attention to technical excellence and good design enhances agility"; According to Cockburn (2002) "A tidy, well encapsulated design is easier to change, and that means greater agility for the project." This principle state the importance of technical excellence which is necessary to move project or changing requirements when needed. When we think about improving quality then we need to do lots of reviewing, testing, and analysis work where we also need to consider about time and cost. Agile methods define technical excellence differently. They focus on more programming practices rather than what programmers develop which means high quality of code is the first requirement. If it is possible to accomplish this work then good quality product will develop automatically where less rework, less retesting and less re-reviews work will be needed. And when changing requirements will be required then it will be easy to handle because of well-structured code (Koch, 2005).

10. "Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely"; this principle points out two main issues, "social responsibility" and "project effectiveness". Employees get tired when they work long hours, their work rate slow down and they start to make mistakes. "An alert and engaged staff is more agile than a tired, plodding staff" (Cockburn, 2002).

11. "Simplicity – the art of maximizing the amount of work not done – is essential"; to achieve technical excellence it is necessary to avoid waste where embracing the philosophy of simplicity the best way. In agile princi-

ple simplicity means "maximizing the amount of work not done" (Koch, 2005).

12. "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly"; this principle states about retrospectives where it also known as "post mortems". A retrospective is a meeting where all project participants discuss on different aspects of project after each iteration. It is almost impossible to get-together all stakeholders after the product has been delivered as they immediately engaged on other project. So following retrospective can help the team to get lesson from each iteration of the project (Koch, 2005).

4.3 Practices and Tools

The principles and values of the agile manifesto may not completely be followed by people or companies who say they follow Agile methods. Individuals may tailor them according to their need and desire. Moreover people may choose to use one of the different kinds of ASD methods or they may combine the different practices from different methods. For this paper we will be focusing on only scrum and XP.

4.3.1 Scrum

Scrum is one type of ASD methodology that is used and developed in the early 1990s by Ken Schwaber and Jeff Sutherland with later collaboration with Mike Beedle. It gets its name from the scrum in rugby and initially introduced by Takeuchi and Nonaka. Scrum values management tools and practices and deals with team level processes to enable the team to work together effectively for good product delivery (Schwaber & Beedle, 2002). Like other ASD methodologies, scrum is an iterative and incremental development method that gives more emphasis to "project management values and practices, rather than those in requirement, implementation and so on" (Larman, 2004). In scrum, the stakeholders play one of the three roles; product owner, scrum master, and scrum team.

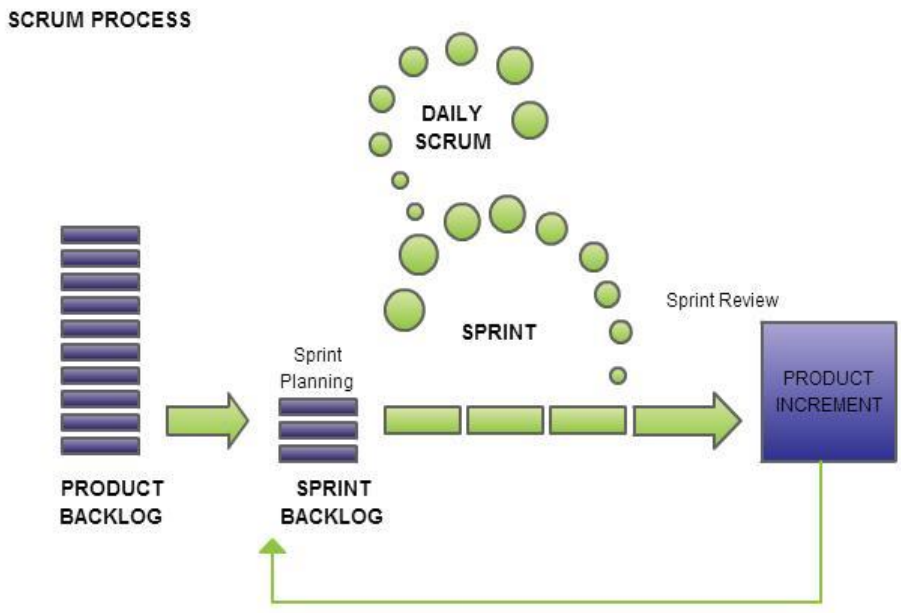

Figure 7: The Scrum process

The Product owner represents stakeholders such as customer, marketing, CEO and so on. The product owner is responsible for creating and prioritizing the product backlog; choosing goals from the product backlog for the next sprint; and are present to review the system at the end of each sprint for sprint review meeting with other stakeholders. There are around 7 core scrum practices that need to be considered carefully while running a scrum project.

The Scrum Master

The scrum master role is usually played by the team leader, project leader, or project manager. Scrum master is responsible for reinforcing the project vision and goals; making sure scrum values, practices and rules are followed by each project stakeholder; mediates between scrum team and management; removes any obstacles and tracks progress; conducts the daily scrum and the sprint review (demo).

Product Backlog

"Product Backlog is an evolving, prioritized queue of business and technical functionality that needs to be developed into a system" write Schwaber & Beedle (2002). It consists of all the functionalities to be built into a system and it's prioritized by the product owner. However every stakeholder in the

project can add items into the product Backlog. The product Backlog grows instantly, but in the beginning for the first sprint to run, it needs to contain enough requirements for the 30 day sprint. The product owner solely control, prioritize and make estimation of how long it would take to develop a feature. To come up with the estimation the product owner works with people (developers, technical writers, quality control staff, and others) who have a strong knowledge of the product and technology.

Scrum Teams

The other role in scrum practice is the scrum team which is played by the development team and they are responsible for working and achieving the sprint goal. The team dedicates all they got to change the requirements of the Product Backlog into a working product. The scrum team is Self-directed and self-organizing team and they have full authority to work on the sprint in any way they choose. The success of a scrum project depends on the scrum team; the team should have dynamics, preferable to have a smaller team size up to 8 people but if there are more, dividing them into multiple teams, cross functional team, a responsible team with full authority, and working environment.

Daily Scrum Meetings

This is a meeting where each scrum team meets to talk about the status of the work they are doing. Daily scrum is a 1 5 minute ritual that the team performs in each working day with all the team members standing in circle and it is lead by the scrum master. During the daily scrum, each team member is asked to answer the three questions: "what have you done since the last scrum?", "what will you do between now and the next scrum?", and "what got in your way of doing work?" Schwaber & Beedle (2002) summarize the advantages of this practice as: "Daily scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve everyone's level of project knowledge."

Sprint Planning Meeting

"Customers, users, management, the product owner, and the scrum team determine the next sprint goal and functionality at the sprint planning meeting. The team then devises the individual tasks that must be performed to build the product increment" write Schwaber & Beedle (2002). The sprint planning meeting consists of two meetings; the first meeting the team with the product owner, management and the users decide what functionality to build during the next sprint; the second meeting is for the team only so that they can decide how to build the functionality. Each 30 day sprint begins with a meeting and the expected outcomes are identifying the Product Back-

log and goal for the next sprint, and defining the sprint Backlog to meet sprint goal.

Sprint
Sprint is the fixed period of time, usually 30 days in scrum that the team works on certain feature of the project in order to come up with an executable product increment. The team already decided the sprint so the task is now to accomplish the sprint goal within the intended time. Within the 30 days sprint, there should be "no interference, no intruders, no peddlers" from the management or other stakeholders. In some circumstances, before the 30 days are over, sprints may get canceled. The reasons could be; the management may change their mind, companies may change direction, and change of market or technology.

Sprint Review
It is a meeting which is coordinated and conducted by the scrum master. "The sprint review meeting is a four-hour informational meeting. During this meeting, the team presents to management, customers, users and the product owner the product increment that it has built during the sprint" states Schwaber & Beedle (2002). The scrum team informs the attendees about the problems they faced and good thing that happened, and the attendees will assess the progress of the scrum team and make in formal decisions about the next step. On this meeting the team also discusses about the plans for the next sprint planning meeting.

Scrum avoids unnecessary and complicated management processes and concentrates on building best product by giving freedom and trusting the team to deliver. Scrum processes seem to be simple and straightforward but still it posses the quality of a proper methodology. It focuses on the effectiveness of the scrum team and creates good communication between the different stakeholders for the delivery of a successful product.

4.3.2 Extreme Programming

Extreme programming, familiarly known as XP, is the most well known Agile method, first publicized by Kent Beck, Ward Cunningham, and Ron Jeffries. XP focuses on the whole team working on a common reachable goal using its values and principles. The team can choose to use appropriate XP practices on their own context. To guide the software development processes XP embraces five values: simplicity (coming up with the simplest design possible today), communication (a good communication between the team

for better performance), and feedback (through constant change there comes feedback), courage, and respect (Beck, 2005). "Values are too abstract to directly guide behavior" argues Beck, but can be supported by different XP principles in order to guide the team's practices in software development. XP's principles can be: humanity, economics, mutual benefit, self-similarity, improvement, diversity, reflection, flow, redundancy, failure, quality, baby steps, and accepted responsibility. However, these principles only give us "a better idea of what the practice is intended to accomplish" writes Beck (2005).

XP has 12 main practices that are well-known and widely used in day-to-day manner in different organizations. But since then there are other practices but we will give the overview of the first practices from (Beck, 2000) and we will include the practice "the whole team" from Beck (2005). "The practices support each other; the weakness of one is covered by the strengths of others." writes Beck (2000).

Planning game

In XP, at the beginning of each development increment there is a planning game that is played in collaboration of the business people (customer, and management) and the technical people in order to "determine the scope of the next release by combining business priorities and technical estimates". The business people need to decide the scope of the problem to be solved, the priority of the features to be developed first and the dates for the releases. The development team provides technical decision that could be considered as a "raw material for the business decisions". They estimate the time for implementation, decide the organization of the work and the team to be involved, provide detailed scheduling of which stories to do first and so forth (Beck, 2000).

Small releases

"Every release should be as small as possible, containing the most valuable business requirements" writes Beck (2000). An XP project team is expected to deliver smaller features frequently, with small-time boxing iterations of few weeks, to the customer. The feature should be something meaningful rather than something that is done for the sake of making the delivery date or partly implemented features in order to make the release cycle shorter. Incase it is not feasible to release the increment for use within the intended date; the increment is demonstrated to the customer in order to show that it is progressing to the intended target.

Metaphors

"Each XP project is guided by a single overarching metaphor" says Beck (2000). The metaphor helps and guides everyone involved on the project

understand the basic elements and their relationships. The metaphor in XP is considered as architecture in other methodologies; it provides the overall concept of the system to be build and it can be easily understood by both the business people and the technical team. It is considered as a high level vision containing the over all system requirements and its expected to remain stable throughout the project (Koch, 2005).

Detailed feature descriptions are stored as stories in a separate document and these can be changed every time if needed. The stories are given short names, graphical descriptions and written on index cards and put on wall. Moreover the stores are estimated very early on their life which can benefit the team to know the value of the feature (Beck, 2000).

Simple design

This practice directly supports XP's value of simplicity which encourages programmers to come up with the simplest code possible rather than designing for the future. XP encourages implementing the simplest design possible today and removing any other complexity discovered later on as soon as possible. Which means "every piece of design in the system must be able to justify its existence" writes Beck (2000). Practicing simple design can have its own advantage of reducing rework than designing for the future.

Test first

XP's test first practice considers before writing any kind of coding, the programmers come up with an automated test first. They write unit tests first and then come up with the implementation of the automated tests in order to verify the story. Moreover, customers write functional tests to make sure it's according to their wish. Test first solves many problems at once: scope creep (help us to focus on writing the code only what the program suppose to do), coupling cohesion ("if it's hard to write a test, it's a signal that you have a design problem, not a testing problem"), trust (teammates trust the author of a code that works), and rhythm (test first restricts the programmer not to get carried away and get lost while coding, there will be a rhythm of test, code, refactor, test, code, refactor). So there will be continuous testing and the tests are run on every program change and this will reduce the time to fix errors (Beck, 2005).

Refactoring

Refactoring is one of XP's well known practice where whenever trying to implement a feature, the programmers take a look at the existing code if they can redesign it to make it simple in order to incorporate the new feature. After adding the feature the programmers still ask themselves if there is a way to make the program simpler. This means that whenever the programmers try to develop a new feature, first they have to revisit the existing fea-

ture and check if they need to redesign. Refactoring helps to remove any duplication, improve communication and add flexibility. After refactoring, all the necessary tests are run in order to make sure that parts that don't need change are intact and the change is implemented successfully (Beck, 2000).

Pair programming

XP has a prominent philosophy of two programmers sitting in one computer side by side and writing all the code. There are two roles to be played between them, one of them with the keyboard and the mouse is responsible for coming up with the best idea to implement that particular method. The other partner would be thinking about other important issues like "if the current approach is going to work?" "Which test cases might not work yet?" "Is there a way to come up with a simplest design to solve the problem?" Beck (2005) sums this practice: "pair programming is a dialog between two people simultaneously programming (and analyzing and designing and testing) and trying to program better. Pair programmers: keep each other on task, brainstorm refinements to the system, clarify ideas, take initiative when their partner is stuck, thus lowering frustration, and hold each other accountable to the team's practices" According to the need, the pairing between the programmers change dynamically.

Collective ownership

In XP project, code ownership is given to every team member since everybody is responsible for the whole system. Anybody who feels the need to change the code, to make an improvement is welcome to do so any where in the system at any time.

Continuous integration

When the pair of programmers complete any kind of code, the change should be integrated and tested within a couple of hours if not possible at most a single day of development. Because "the longer we wait to integrate the more it costs and the more unpredictable the cost becomes" writes Beck (2005). If the tests could not run 100%, the pair should throw away what they did and try to come up with a new one and repeat the same procedure again.

Energized work

ASD has the principle known as sustainable pace where it talks about how the team members, the sponsors and users should be able to maintain a constant pace indefinitely. As one of the well-known method of ASD, XP agrees on limiting the employees working hour in a week to 40 hours. In XP over time is considered as a serious problem on the project. One week of overtime can be tolerated but a second week of overtime in a row is not an

option in XP project. People that are energized and fresh can be more productive on their work than those who didn't get much rest.

Onsite-customer

XP encourages the involvement of real customers available with the team to answer questions, resolve disputes and to set priorities. The customer should be one who uses the system when it is completed, one whose lives and business is affected by the system and one who we try to please (Beck, 2000 & 2005). The more we involve the customer and give priorities to the needs of the customer the better the development outcome would be. The disadvantage of involving customer arises if the project gets canceled and all the time the customer spent would be for nothing. However, XP does everything possible to make sure that doesn't happen.

Coding standards

It would have been impossible to control "who on the team wrote what" code in XP, if there isn't a practice called coding standards. Because in XP anybody can change the code at any time, programmers swap partners a couple of times in a day, and they refactor each other's code constantly so it would have been difficult if there are different coding practices. Coding standards make these practices possible with everyone following the same type of coding practices. The standard emphasizes communication and most importantly, "it must be adopted voluntarily by the whole team" writes Beck (2000).

The whole team

Beck incorporates this practice of XP underlying how it is important to include cross-functional teams with all the skills and perspectives working together in interlinking ways for the success of the project. People in the team need to have the sense of belonging, that they are in this together and that they support each others' work, growth, and learning writes Beck (2005). Beck explained the different stakeholders that are involved in the XP team: testers (help customers choose and write automated system-level tests and coach programmers on testing techniques); interaction designers (choose over all metaphors for the system, write stories, and evaluate usage of the deployed system to find opportunities for new stories); architects (look for and execute large-scale refactoring, write system-level tests that stress the architecture, and implement stories); project managers (facilitate communication inside the team and coordinate communication with customers, suppliers, and the rest of the organization); product managers (write stories, pick themes, stories in the quarterly cycle, pick stories in the weekly cycle, answer questions); executives (provide an XP team with courage, confidence and accountability, sponsoring or overseeing XP teams); technical writers (provide early feedback about features and to create close relationships with

users); users (help write and pick stories and make domain decisions during development); programmers (estimate stories and tasks, break stories into tasks, write tests, write code to implement features, automate tedious development process, and gradually improve the design of the system); human resources (reviewing and hiring employees).

4.4 Summary and Definition

More than a decade has gone since the world of information systems development has seen the emergence of Agile methods. The most popular ones includes; eXtreme Programming (XP), the Dynamic Systems Development Method (DSDM), Scrum, Crystal and Feature Driven Design. There is no common definition for Agile methods since various practices exist but one can be sure on the definition of Agility. According to Conboy (2009) Agility means "the continual readiness of an Information Systems Development method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment." Agile methods are guided by sets of values that give priority to Individuals and interactions over processes and tools, working software over comprehensive documentation, Customer collaboration over contract negotiation, and Responding to change over following a plan. These set of values are broad and further divided into core principles. The principles include customer satisfaction, delivering working software frequently, embrace change, simplicity, stakeholder involvement, motivated and self-organizing teams, sustainable development, and technical excellence. These principles applied jointly can create organizations with higher business value out come, "motivates employees, encourages innovation and delivers tangible results" (Cooke, 2010). At some occasions the different Agile methods follow their own way of achieving Agility. For example XP has a practice called collective code ownership while Feature Driven Design demands individual code ownership. This contradiction can be challenging and confusing for those who want to be Agile, writes Conboy (2009).

XP and Scrum are the two most commonly known Agile methods ("commercial Agile methods") that organizations use these days. One may argue that the popularity of XP also paved the way for the emergency of similar methods. XP is a "lightweight" development process but also extreme in that it "takes commonsense principles and practices to extreme levels" states Beck (2000). XP is based on values of communication and feedback between developers and customers, the art of simplicity, and responsibility of

the team. XP consists of well known processes and practices that have been used by developing teams. These includes planning game, small releases, metaphors, simple design, test first, refactoring, pair programming, collective ownership, continuous integration, energized work, onsite customer, coding standards, and having the whole team together. One may pursue to argue that XP is well suited for small to medium sized projects. However others have suggested that XP's best practices like small releases can help large projects to downsize.

Scrum is an Agile software development methodology dedicated to project management. It deals with team level practices and processes. In Scrum, the team plays one of the three roles; Product owner, Scrum Master, and Scrum team. The Product owner is responsible for creating and prioritizing the Product Backlog and also be present for the Sprint review meeting. Scrum Master is the engine of the project. He or she is responsible for making sure Scrum values, practices and rules are followed by the Scrum team. Moreover this person is responsible for removing any obstacles and tracks progress in addition conducts the daily Scrum and the Sprint review meetings. The Scrum process is comprised of daily Scrum meetings, the 30 days Scrum Sprint for the next product increment, Sprint planning meeting to decide the next product to be developed, Sprint review demonstration, and processes for managing the Sprint Backlog and Product Backlog.

5. Lean methods

5.1 Introduction

"Lean" philosophy has been described in different branches of application domains ranging from manufacturing systems to organizations where applications domains could be "Lean Manufacturing" (LM), "Lean Production" (LP), "Lean Thinking" (LT) etc (Putnik, 2012). In this thesis we are interested in organizational domain where application domain considered as Lean manufacturing.

Lean has been adapting for different companies from many years ago, however the success of Japanese industries in automotive area created a new revolution for lean methods. One of the lean philosophies "Toyota Production System" (TSP) has been described in many books. Besides, "The Toyota Production System: Beyond Large-Scale Production" by Ohno (1988), "Lean thinking" by Womack and Jones (2003) are well-known book for lean methods. Moreover, several well-known authors like Poppendieck and Poppendieck (2003), Middleton (2001), Middleton, Flaxel & Cookson (2005), Holden (2010), Åhlström and Karlsson (1996, 1998), Poppendieck and Cusumano (2012), Liker and Morgan (2006) published different articles based on lean methods.

The "Lean" methods are not new rather it is becoming crucial for today's manufacturing companies for long-term survival. Increasing global competitions, insufficient resources and unpredictable economies could be the reasons of adapting lean in different organizations. Scherrer-Rathje, Boyle & Deflorin (2009) defined "Lean is a management philosophy focused on identifying and eliminating waste throughout a product's entire value stream, extending not only within the organization but also along the company's supply chain network". Lean helps organizations to decrease their expenses by eliminating wastes, improving qualities and also increase customer satisfactions (Moayed and Shell, 2009). Lean also reduces delivery time and increase product improvement where evidence could be collect from different car companies like Toyota (Japan), Porsche (Germany) and Pratt & Whitney (USA) (Middleton, 2001 cited on Womack and Jones, 1997). It is also true that adapting lean in the organization is not too easy, for a success-

ful project it requires a clear perception of lean methods. European manufacturer of food processing equipment launched lean methods in 1997 which was failed and later in 2006 it succeed (Scherrer-Rathje, Boyle & Deflorin (2009).

A very popular and well-known difference between mass production and lean production is defined by Womack and Jones in 1997 ie. "Perhaps the most striking difference between mass production and lean production lies in their ultimate objectives. Mass producers set a limited goal for themselves—"good enough," which translates into an acceptable level of defects, a maximum level of inventories, a narrow range of standardized products. To do better, they argue, would cost too much or exceed inherent human capabilities. Lean producers, on the other hand, set their sights explicitly on perfection: continually declining costs, zero defects, zero inventories, and endless product variety" (Middleton, 2001 cited on Womack and Jones, 1997).

Of course lean come from manufacturing areas though it is possible to obtain enormous advantage by applying lean in software development areas. But it is also necessary to distinguish between manufacturing and product development. Manufacturing can be defined with predictable and repetitive tasks with a large numbers of delay costs and standardized task. On the other hand, product development can be defined with high variability, non-repetitive, non- homogeneous flows, variable delay cost with variable resources load (Rudolf and Paulisch, 2010 cited on Reinertsen n.d.).

According to Middleton and Joyce (2012) software development is relatively more acquiescent and cheaper that manufacturing products. However, the idea of applying principles behind the development is still same for example in software development using Kanban is a fruitful result.

In manufacturing area tasks can be classified into three categories: incidental work, value-added work and muda. Incidental tasks are those tasks that do not add value to the product though it required satisfying other product. Value-added processes are those that directly related to product or we can just say that creates value to the customers. Non-value added process or mudas are those that add no values in the product (Chena, Li & Shady, 2010 cited in Monden, 1998). Mudas can be classified into seven categories like overproduction, waiting, transportation, over processing, inventory, motion and defects (Ohno, 1988; Womack and Jones, 2003). Poppendieck and Poppendieck (2003) have moderated those seven wastes in software development as partially done work, extra processes, extra features, task switching, waiting, motion and defects. To identify and decrease the amount of muda lean tools Kaizen is the best way to achieve this where Kaizen helps to employees in

thinking small improvement ideas on a regular basis (Chena, Li & Shady, 2010).

5.2 Lean principles

Lean practices have been adapting in manufacturing companies more than for the last 60 years. Different authors provide different lean principles rather we can conclude that lean has no pure principles. In this section we are going to describe some famous author's mentioned principles that are possible to apply in software development.

Ohno (1988) described elimination of waste is the first principles in Toyota production system. Ohno (1988) identified seven sources of wastes are: "waste of overproduction, waste of time on hand (waiting), waste in transportation, waste of processing itself, waste of stock on hand (inventory), waste of movement and waste of making defective products". *Eliminating these wastes* is one way to improve product efficiency. Another principle is *built-in-quality* which refers to produce good quality product to minimize defects. To build good quality product Ohno (1988) talked about "*automation*" i.e. give the machine intelligence. The significance of automation can be identified in two ways. One way is that it can reduce overproduction and another way is that it can prevent producing defective products. Another principle is slow growth production. Ohno (1988) suggested decreasing the concept of mass production where it creates different kind of wastes. *Flow of process* is another principle where JIT method is used to establish this principle. JIT is only the required parts will be produced. Ohno (1988) talked about *cost reduction*. He suggested to abandon conventional profit calculation (selling price= profit + actual cost) rather to think about the value of product to the market and market competition. To increase production efficiency it is necessary to *empowering team* as well as creating multi-skill operator. It will help to reduce labor cost and depreciation burden. Another principle is *production leveling*. It establishes a production flow through maintaining a constant supply of products. The other principles described by Ohno (1988) are *work standardization*, *solve the problem from root cause* (identify source of the problem).

Poppendieck and Poppendieck (2003), on their book called "Lean Software Development: An Agile Toolkit", discussed the application of lean principles to software development. They talked about the different processes of translating lean principles to agile practices to individual software development domains. While doing that, they have emphasized on seven lean principles, which are: *Eliminate waste*: in lean thinking, anything that doesn't

add value to a product and that doesn't satisfy the customer's need is considered as waste; *Amplify learning*: in every iteration, there can be a new practice or learning to be gained, since we don't expect perfect work within the first delivery; *Decide as late as possible*: rather than making early decisions and prone to error, it's better to decide late after getting enough information for practices that involve uncertainty; *Deliver as fast as possible*: though speedy delivery customers can get "what they need now" rather than "what they needed yesterday" and moreover, it also helps to facilitate the learning process when deliver cycles are shorter; *empower the team*: involving workers in key decision making so that they know what need to be done by themselves and this is performed by using *pull* mechanism; *Build integrity in*: delivering a system that exactly matches and fulfills the user's wishes and needs; "Software with integrity has a coherent architecture, scores high on usability and fitness for purpose, and is maintainable, adaptable, and extensible." ;and *See the whole*: implement deep expertise in every system area to enhance the overall system performance.

Womack and Jones (2003) on their book, "Lean thinking: Banish waste and create wealth in your corporation" defined lean manufacturing as a five step process: defining customer value, defining the value stream, making it "flow", "pulling" from the customer back and striving for excellence. After interactions with many audiences and considerable reflection, the authors concluded that lean thinking can be summarize in five principles: *"precisely specify value by specific product", "identify the value stream for each product", "make value flow without interruptions", "let the customer pull value from the producer" and "pursue perfection"*. Identifying *value stream* means to find out the sequence of processes from supplier to customer while developing a product. *Creating flow* means to make the flow of values and ignore making any delay while value-adding activities and develop product "once-at-a-time". *Pull products means* only develop the required products required by customer. Finally *perfection* is to improve the system continuously by reducing waste (eliminating effort, time, space and errors) while development.

Middleton (2001), while discussing how the concept of lean manufacturing can be successfully applied to software development, mentioned 5 principles of lean manufacturing as Schonberger, R. J. on this book, "World Class Manufacturing", 1986 put it: *Continual quality improvement, project after project; Empowered workers taking responsibility; Defect prevention, not random detection; Simple, visual measures of quality and Automatic quality measurement devices, often self- developed.*

Middleton, Flaxel & Cookson (2005), on their paper on "Lean Software Management Case Study: Timberline Inc." have performed a case study on

full adoption of lean by a software company, the case study elaborate and emphasize on how lean thinking can work successfully for software developers. To do that, the team members applied the different principles and techniques of lean thinking for their software processes, which are: *Continuous-Flow Processing:* instead of waiting for several months or years to release the whole product, it's better to handle inventory in a smaller batches*; Customer Defined Value; Design Structure Matrix (DSM) and Flow:* this has to do with breaking down the requirements so that the work to be done can be estimated accordingly*; Common Tempo or 'Takt' Time*: "to pace work according to customer demand" *; Linked Processes*: linking of processes with their components by putting them closer *; Standardised Procedures*: e.g. file storage, names, locations, workspace, will help people to move from one project to another*; Eliminate Rework; Balancing Loads; Posting Results; Data Driven Decisions;* and *Minimise Inventory*.

Carvalho, Alves & Lopes (2011), on their work about the principles and practices of lean production applied in a metal structures production system, emphasized the use of lean production and some lean tools to improve the production process and solve production problems that occurred such as: deliveries delays, long lead times, too many material handling, high stocks, errors and defects in metal structures assembly and production, and unnecessary motions. On this article they talk about some principles of lean thinking that are applied for this specific domain, which are: *create value for the customer*, *map the value stream*, *create flow*, *pull the production* and *pursuit perfection*.

Highsmith (2002), in his book called "Agile software development ecosystem", puts Lean development under the ecosystem of Agile software development and emphasizes how Lean development is derived from the principles of lean production. Highsmith summarizes 12 principles of Lean development as: *satisfying the customer is the highest priority*: In lean thinking customer is given a higher priority and not satisfying the customer is considered as a failure; *always provide the best value for the money*:"Value is a combination of product features that meets a customer's needs at a specific time for a specific price." ; *success depends on active customer participation*: there must be active customer collaboration to make early changes and decision making ; *every Lean development project is a team effort*: "multidisciplinary teams rather than isolated individuals are needed because diversity is key to innovative, fast-cycle time development." ; *everything is changeable*: there is a continuous change in requirement so adapting to change instead of trying to control it ; *domain, not point, solution*: developing software that is applicable to multiple domains can reduce the cost and and increase value; *complete, don't construct*: ; *an 80% today instead of 100% solution tomorrow*; *minimalism is essential*: "by minimizing paper-

work, keeping teams small and collocated, and keeping the product scope focused" Lean development eliminates waste ; *needs determine technology*: first priority should be given to the application development then we can choose the technology to support it; *product growth is feature growth, not size growth*: instead of the size of the product, emphasis should be given to delivering "change -tolerant features"; and *never push Lean development beyond its limits*: identify only problems that lean development designed to handle.

Larman and Vodde (2009), on their book, "Lean Primer" has discussed about 14 lean principles that originally have been described in the Toyota way book. The principles are: *management decisions should be based on long term philosophy, move toward flow, use pull systems, "reduce variability and overburden", solving problems in the middle, "master norms", "use simple visual management", use proven technology;9. Create leaders; 10. "Develop exceptional people", "help partners be lean", going and checking on in person, careful decision making, and create learning environment through "relentless reflection and kaizen".*

Lean is a combined method where it focuses on every characteristics of an organization specially on developing people. Liker described fourteen lean principles where seven principles have been followed in BBC software development environment. *Levels of WIP, pull work, level out the workload, build a culture of stopping fixing problems, continuous improvement, visual controls and use of technology* were considered as Lean software development principles. *Level of WIP* is refers that the requirements, designs and coding stuffs should be keeping as low as possible when need to deliver. It helps to create a continuous flow while developing and bring out problems if occurred. *Pulling work* refers to assign work on available capacity rather than pushing work. *Leveling work* helps to predict and estimate project. *Building a culture of stopping fixing* problem will increase product reliability and efficiency. *Visual controls* are necessary because it creates a transparent environment while developing products. Similarly it is also necessary to take care about the use of technology (Middleton and Joyce, 2012).

Holden, (2010) identified key Lean principles as: *eliminate waste to maximize value to the customers, continuous work flow with minimum delay, worker empowerment, autonomation, problem solved from their source and continuous improvement*.

Åhlström and Karlsson (1996, 1998) have described eight lean principles for product development. *Elimination of waste* is the first principle of lean production philosophy. Waste is something that has no value to the customer where the most important source of waste is inventory. *Continuous im-*

provement is to improve production system gradually where perfection is the only goal. *Zero defects* refer how to achieve high quality. To achieve high quality it is necessary to concentrate on all parts and products error free from the beginning of production. Another principle is *pull scheduling* which refers product is listed through a pull system (customer order). Multifunctional team refers to those teams or a group of workers whose can operate different tasks. Statistic showed that a *multifunctional team* can perform much better than any traditional work organizations. *Delayering or decentralized responsibilities* is another lean principle where there is no supervisor and multifunctional team is likely to perform supervisory tasks. It also helps to reduce hierarchical levels in the organization. *Team leader* is another principle which carries all of the supervisory roles like advisory, coaching, providing support etc from multifunctional team. And finally the last principle is *vertical information system* which refers that information flows directly to the relevant decision-makers. It helps instance feedback and corrective action.

Recently Poppendieck and Cusumano (2012) have described their modified seven lean principles for software development which is based on subsequent experiences. Authors suggested thinking lean as a set of principles rather than thinking a set of practices. Then applying lean principles in software development make more sense and increase product quality. The lean principles for software development are: *optimize the whole, eliminate waste, build quality in, learn constantly, deliver fast, engage everyone and keep getting better*. *Optimize the whole* is to identify customer's needs and what they will value. Authors identified unnecessary features, lost of knowledge, partially working, handovers and multitasking, 40-50% of development time spent for finding and fixing bugs are the wastes in software development. On the other hand *continuous integration* is the best way to build good quality software. *Learning* can be achieved in two ways: one is "learn first" approach which is to traverse multiple operations like fundamental architecture, choice of language, design language for user interaction etc and take decision as late as possible. Another approach is to build the basic product and update it by frequent delivery from customer feedback. For product delivery authors suggested Harlan Mills's idea. Rather than thinking software development as a project, it is a good idea to think software as a flow where design, development, delivery etc are flow of small changes. Authors suggested thinking as a product development rather than thinking software development and it includes the people who have knowledge on customers, designers, developers, testers, operations, support and finance. And finally there is no end of doing best. So principles suggested that to *improve work* constantly over time by the people and teams.

Liker and Morgan, (2006) have described thirteen lean product development principles. These principles are grouped in three areas: process principles;

people principles; tools and technology principles. Process principles group contains *establishing customer value; decide as late as possible* while traversing all possible alternative solutions; *creating level for process flow* and use *standardization* to reduce variation. Best way to *establishing customer value* is by eliminating waste. Waste is that take time, money and resources and at the end does not have any value from customer's side. *Creating a leveled of product* development process flow helps to predict and estimate project. *Standardization* is one of the principles which is the fundamental of continuous improvement. Toyota use three categories of standardization like design standardization, process standardization and standardized skill sets. People principles group contains: *developing a "Chief Engineer System"*; a *balance functional expertise and cross-functional integration; developing towering technical competence in all engineers; involving suppliers into product development; building learning environment with continuous improvement and building a culture to support excellence*. *A chief engineer* is that character who will be responsible from the start to end of the product development. Toyota distinguish chief engineer from project manager. A chief engineer is one who acts as a project manager, manage people and time and act as a chief technical architect when needed. Team should be *cross functional* where Toyota encourages following matrix organization structure. And finally tools and technology principles group contains: *adapting technology to fit employees; align organization through simple and visual communication and finally use powerful tools for organization learning.*

Ebert, Abrahamsson & Oza (2012) describe the five core lean principles as: *customer focus, waste reduction, team empowerment, work stream efficiency and continuous improvement.*

Based on the above author's descriptions we have summarized the following leans principles:

Table 1: Lean principles mentioned by different authors

Authors	Lean principles
Åhlström and Karlsson (1996, 1998)	Elimination of waste (ÅK1); Continuous improvement (ÅK2); Zero defects (ÅK3); Pull scheduling (ÅK4); Multifunctional team (ÅK5); Delayering or decentralized responsibilities (ÅK6); Team leader (ÅK7); Vertical information system (ÅK8).
Ebert et all (2012)	Customers focus (E1); Waste reduction (E2); Team empowerment (E3); Work stream efficiency (E4); Continuous improvement (E5)
Highsmith (2002)	Satisfying the customer is the highest priority (H1); Always provide the best value for the money (H2); Success depends on active customer participation (H3); Every Lean development project is a team effort (H4); Everything is changeable (H5); Domain, not point, solution (H6); Complete, don't construct (H7); An 80% today than a 100% solution tomorrow (H8); Minimalism is essential (H9); Needs determine technology (H10); Product growth is feature growth, not size growth (H11); Never push Lean development beyond its limits (H12).
Holden, (2010)	Eliminate waste (HO1); Continuous work flow with minimum delay (HO2); Worker empowerment (HO3); Autonomation (HO4); Problem solved from their source (HO5); Continuous improvement (HO6).
Larman & Vodde (2009)	Management decision on long-term (LV1); Flow (LV2); Pull systems (LV3); Reduce variability and over burden (LV4); Adapt stopping and fixing problems (LV5); Master norms (practices) (LV6); Simple visual management (LV7); Well-tested technology (LV8); Leader-teachers from within (LV9); Develop exceptional people (LV10); Help partners be lean (LV11); Make decisions slowely by consensus (LV12); Go see (LV13); Reflection & kaizen (LV14).
Liker and Morgan, (2006)	Establishing customer value (LM1); Decide as late as possible (LM2); Creating level for process flow (LM3); Standardization (LM4); Developing a "Chief Engineer System" (LM5); Balance functional expertise and cross-functional integration (LM6); Developing towering technical competence in all engineers (LM7); Involving suppliers into product development (LM8); Building a culture to support excellence (LM9); Building learning environment (LM10); Adapting technology to fit employees (LM11); Align organization through simple and visual communication (LM12); Use powerful tools for organization learning (LM13);

Authors	Lean principles
Middleton (2001)	Continual quality improvement, project after project (M1); Empowered workers taking responsibility (M2); Defect prevention, not random detection (M3); Simple, visual measures of quality (M4); Automatic quality measurement devices, often self-developed (M5).
Middleton, et al (2005)	Continuous-Flow Processing (MEA1); Customer Defined Value (MEA2); Design Structure Matrix (DSM) and Flow (MEA3); Common Tempo or 'Takt' Time (MEA4); Linked Processes (MEA5); Standardised Procedures (MEA6); Eliminate Rework (MEA7); Standardised Procedures (MEA6); Eliminate Rework (MEA7); Balancing Loads (MEA8); Posting Results (MEA9); Data Driven Decisions (MEA10); Minimise Inventory (MEA11).
Ohno (1988)	Waste elimination (O1); Built-in-quality(O2); Automation (O3); Slow growth production (O4); Process flow (O5); Cost reduction (O6); Empowering team (O7); Multi-skill operator (O8); Production leveling (O9); Work standardization (O10); Identify source of the problem (O11); Pull scheduling (O12).
Poppendieck and Poppendieck (2003)	Eliminate waste (P1); Amplify learning (P2); Decide as late as possible (P3); Deliver as fast as possible (P4); Build integrity in (P5); See the whole (P6); empower the team (P7).
Poppendieck and Cusumano (2012)	Optimize the whole (PC1); Eliminate waste (PC2); Build quality in (PC3); Learn constantly (PC4); Deliver fast (PC5); engage everyone (PC6); Keep getting better (PC7).
Womack and Jones (2003)	Defining customer value (WJ1); Defining the value stream (WJ2); Making it "flow" (WJ3); Pulling" from the customer back (WJ4); Striving for excellence (WJ5).

5.3 Practices and Tools

Lean methods have numerous numbers of practices that have been adapting in manufacturing companies. Moreover, in Toyota Production System (TSP) Just-in-time (JIT), Jidoka, Heijunka and Stable, Standardized Processes and Kaizen are very famous practices. In this paper we have included only those practices that are directly related to software development and few of them are discussed below:

5.3.1 Just-in-time

Just-in-time (JIT) is one of the most popular lean manufacturing practices for the last three decades where practicing JIT in software development might be beneficial. JIT can be defined as "*to produce the necessary units in the necessary quantities at the necessary time" (Yavuz, 2010 cited in Monden, 1998, p.59).* It is the goal for all companies to reduce the use of all key resources like labor, capital, materials, space and time. JIT uses several branches of tools for example pull approach and *Kanban* production control, inventory reduction, quick setups and orders, quality at the source (jidoka), supplier networks, teamwork and participation, continuous improvement (*Kaizen*) that can be use to achieve the above efforts (Bruuna and Meffordb, 2003).

Ohno (1988) mentioned "Kanban is a way to achieve just-in-time". It helps the employee to start their work by themselves and take self decisions when needed. *Kanban* also mentioned what are the roles and responsibilities for the managers and supervisors. Eliminating waste is the main purpose of lean manufacturing where *Kanban* visualize wastes. There are different sources of wastes in software development for example partially done work, extra processes, extra features, task switching, waiting, motion and defects (Propendieck, 2003). Even *Kanban* also helps to reduce manpower and inventory. Different manufacturing principles suggested that rather than pushing work use pulling method. *Kanban* is the best way to achieve pull system.

5.3.2 Kaizen

Kaizen is one of the well known tools in lean manufacturing where it has been used for continuous improvement (Barraza, Smith & Dahlgaard-Park, 2009; Holden, 2010; Wanga, Conboyb & Cawleyc, 2012). Kaizen is a Japanese word where it means "continuous improvement" ("kai" means "change" or "to correct" and "zen" means "good"). Continuous improvement could be concern on varies areas of the organization for instance customer focus improvement, product quality improvement (one way is by re-

ducing process cycle time), reducing production cost and as well as improving delivery performance and customer satisfaction (Barraza, Smith & Dahlgaard-Park, 2009 cited on Sohal, 1996) As these improvements are directly related to software development so applying Kaizen in software development might be advantageous.
One of the popular lean principles developed by Womack and Jones (2003) suggested five steps for lean manufacturing: value, value stream, flow, pull and perfection. Ikuma, Nahmens & James (2011) believe that these principles can be implemented through *Kaizen*.

5.3.3 Queuing Theory

Queuing theory is a mathematical method which is used for calculating delay of waiting in a line. Propendieck (2003) suggested brief description on queuing theory not only helps for addressing the problem but also ensures deliver products fast. The fundamental measurements of queuing theory are: cycle time, rate of arrival, rate of service.

- Cycle time:" Cycle time is the average time it takes something to get from one end of a process to the other" (Propendieck, 2003). On the other way we can say cycle time is the sum of queue time and service time. For example if we want to calculate cycle time from the entrance of an airport to the gate then cycle time will be the sum of time spent in for luggage checking, time checking for luggage and getting boarding pass, waiting in the security line, time to get through security and time it takes to walk to the gate. The more we can reduce cycle time the faster we can deliver product. There are two ways of reducing cycle time: rate of arrival and rate of service.
- Rate of arrival: Rate of arrival is considering the way arrival works where spreading out arrivals demand decreases cycle times. It is necessary to keep fixed rate of arrivals which ensure shorter cycle time. (Propendieck, 2003) suggested that releasing small packages of work ensure controlling the rate of arrival work.
- Rate of service: Besides arrival rate of service it is also necessary to remove variability from the processing time. Unlike arrival, small releasing packages also increases service rate. Small releasing packages also allow performing parallel small works where if anyone is stopped for any reason others can proceed without delay.

5.3.4 Leadership

Leadership is one of the key practices in lean manufacturing organizations. Most of the authors showed the importance of leadership in lean manufacturing organizations (Ohno, 1988; Larman & Vodde, 2009; Åhlström and Karlsson1996, 1998; Liker and Morgan, 2006; Poppendieck and Poppendieck, 2003). Sometimes people mix managers and leaders but in reality these two are not same at all. The role for managers could be planning and budgeting, organizing and staffing, tracking and controlling where the role for leaders could be setting direction, aligning people ie team members and motivate the team (Poppendieck and Poppendieck, 2003).

.

Normally a company may have different departments where each department has its own roles and responsibilities. No one need to take care on each other's working status, then question arise where people should knock for available working status or where decisions are made? Toyota creates a character known as "Chief engineers" to face such situations. Poppendieck and Poppendieck (2003) mentioned the roles at Toyota for "Chief engineers" are studying market, writing documentation for vehicle, establishing several designs, setting schedule and responsible for economic performance of the vehicle. At the end Liker and Morgan (2006) showed the importance of leadership by asking the following question:

"Who is responsible for taking product or service from start to finish with the deep expertise to see it is all done effectively with a high degree of expertise?"

5.3.5 Cost of delay

Cost of delay is brief calculation of cost and profit before taking any decision. For example in software organization developers may ask manager to buy or develop new tools to make their work more efficient and faster. Rather than just accepting their request, cost of delay suggested calculating cost and profit for the new tool first. Conventional theories suggested that managers should agree or disagree based on the profit/lost calculation. However, Preston Smith and Donald Reinertsen published a book named "Developing Products in Half the Time" suggested that profit may larger than theoretical calculation. A detail example of cost of delay calculation is described by Poppendieck and Poppendieck (2003) in their book "Lean Software Development: An Agile Toolkit".

5.3.6 Value stream mapping

Value stream mapping is one of the strong lean practices proposed by Womack and Jones (2003). Identifying value stream means to find out the sequence of processes from supplier to customer while developing a product. Poppendieck and Poppendieck (2003) also suggested using value stream

mapping in software development which could be a strong tool for eliminating waste. To achieve value stream mapping tools could be only paper and pencil where managers are encourage to visit the overall process from the beginning to end of a project. While visiting the overall process managers should draw a chart which shows all the necessary steps including timeline (timeline shows the required time to complete a specific step). It also helps managers to draw a timeline for waiting states that also helps to eliminate waste ie eliminating non-value adding activities.

5.4 Summary and Definition

Historically, "Lean" method is not a new word rather it has been using in manufacturing companies for the last few decades. Putnik and Putnik (2012) mentioned, "However, the concept came to attention after the business success of Japanese industry especially in the automotive sector". Many organizations are now trying to adapt lean method in order to achieve its profitable result as well as to keep them alive in this competitive local/global market. By eliminating waste, improving quality and increasing customer satisfaction lean helps the organizations to decrease their costs (Moayed and Shell, 2009). And finally Conboy (2009) defined leanness as "contribution to perceived customer value through economy, quality and simplicity".

In literature, Lean practices have been introduced more than 60 years ago where there are no pure lean practices or principles yet. Different scholars provide different principles where they explain principles in different way for example Ohno (1988) has twelve principles, Womack and Jones (2003) have five principles, Larman & Vodde (2009) have fourteen principles, Liker and Morgan, (2006) have thirteen principles, Åhlström and Karlsson (1996, 1998) have eight principles. Besides industrial (automotive) sectors currently lean has also affected software organizations which increase product improvement. Many researchers are now encouraging adapting lean methods in software development. Although all the lean principles are not applicable for software development, researchers are tailoring lean principles for software development. For software development Poppendieck & Poppendieck (2003) delivered seven lean principles and then later on 2012 they have modified those seven principles on Poppendieck and Cusumano (2012), Highsmith (2002) derived twelve principles, Middleton, et al (2005) derived eleven principles. See section 5.2 for details. Although, sometimes it is difficult to distinguish between principles and practices, to achieve the lean principles different practices were suggested by different authors for software development are listed below:

Lean software development practices	
Just-in-time (JIT)	Yavuz, 2010 cited in Monden, 1998, p.59; Ohno, 1988
Kanban	Ohno, 1988; Polk, 2011
Kaizen: continuous improvement to establish smooth flow	Ohno, 1988; Barraza, Smith & Dahlgaard-Park, 2009; Holden, 2010; Wanga, Conboyb & Cawleyc, 2012
Jidoka: quality at the source. Automation, machine should serve for the people only	Bruuna and Meffordb, 2003
Defer decision making	Poppendieck & Poppendieck, 2003
Kano analysis: increase customer satisfaction	Miskelly, 2009
Make everything visible	Womack and Jones, 2003; Middleton, 2001
Queuing Theory: to measure and manage activities	Propendieck and Propendieck, 2003
Cost of delay: to calculate cost	Poppendieck and Poppendieck, 2003
Value stream mapping: analysis and design the workflow in order to deliver the product to the customer	Womack and Jones, 2003; Poppendieck and Poppendieck 2003
Heijunka: leveling workload, smooth production, reducing muda	Middleton, et al 2005; Wanga, Conboyb & Cawleyc, 2012

6. Derived Agile and Lean principles

In this chapter we are going to derive both Agile and Lean principles based on literature review. For Agile principles we are going to use Agile manifesto directly as Agile manifesto principles are mostly followed by most of the software companies. And for Lean principles we are going to use different author's suggested principles (principles those are only related to software development). Moreover we are also going to identify relevant practices and stakeholders under each principle to make principles more understandable.

6.1 Agile principles based on stakeholder perspective

In software development "Agile" is a well-known name where most of the software companies using agile methods directly. As there are not too many argues in agile principles therefore, we are going to derive the following agile principles based on Agile Manifesto.

6.1.1 Working software frequently

Delivering working software frequently from a couple of weeks to a period of one month of time has been the special signature and one of the primary principles of ASD. Two principles are under this heading ("Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale" and "Working software is the primary measure of progress"). These principles are derived from the second value of the Agile manifesto "working software over comprehensive documentation". ASD values delivering a working code in shorter time boxed iteration to the customer instead of plans and too much documentation. ASD breaks down the project into smaller size functionality systems so that it can be possible to come up with a working code within the different iterations and this will play important role towards the customers' satisfaction. ASD emphasizes in its principle how working software should be the major progress on the project than a bunch of organized plans and scheduled documents.

Each Agile method has practices that support these principles. However since Scrum and XP are only covered in this paper, we will see practices from the two only. Out of 12 principles in XP two of them (*small releases* and *continuous integration*) support the principle of delivering working software frequently. In small releases, XP encourages the smallest release possible with important features that can bring value to the customer included. Delivering small releases frequently will allow the customer to give relevant feedback early which at the same time helps the team to include it on their next release plan. XP's continuous integration is an ongoing process performed by the pair of programmers, after finishing any code they integrate it to the product. This process even makes the feedback faster because it is performed everyday at any time (Highsmith, 2002).

In Scrum, frequent delivery is discussed under the practices of the *sprint* and the *sprint review*. A sprint is the small release that the team works on to deliver in the next 30 days of period. The sprint team uses that time to work on the stories that are going to be implemented with complete control and freedom. The team can use whatever strategies or methods to complete the sprint on the specific time box. When completed, each sprint is presented and reviewed on the day of the sprint review in the presence of the product owner, the customer, management, the scrum master and other project stakeholders. The sprint review helps every project stakeholder to learn from the sprint that is already developed and to plan a session for the next sprint to be developed (Koch, 2005).

- **Identified practices/tools**: XP (*small releases and continuous integration*), Scrum (*sprint and sprint review*).
- **Identified stakeholders**: Programmers, customers, requirement analyst, designers, testers, scrum master, product owner, management (senior manager, middle-management, project manager) and users.

6.1.2 Embrace change

Welcoming change of requirements is considered as a motto for ASD. It is derived from the fourth value of the agile manifesto, which is "responding to change over following a plan". ASD welcomes changing requirement even late in the development process to increase the customer's competitive advantage. Organizations which embrace change in their customer's requirement, market demand and available technology have a better competitive advantage and customer satisfaction than those organizations that are less

responsive to change. Embracing change is one of the significance differences between ASD and other traditional methodologies, which controls change instead. Because these plan driven methodologies consider change to be extra cost unlike ASD.

Adaptive software development, DSDM, FDD, XP, and Scrum are the well known Agile methods that posses practices that directly support this principle. The two famous practices of XP which talk about changing requirements are *metaphor* and *refactoring*. Metaphor describes what the project is trying to achieve. It describes what kind of product the project is trying to produce in broad sense. These can create some kind of coherent for the team and in some extent for the customers too. It serves as a principal document for the project stakeholders in order for them to keep track of the goal of the project, at the same time welcoming requirement changes. Beside metaphor XP has another element called *stories*, which holds the detailed requirements in the project that are designed to welcome change. Every time new stories can be added or deleted or modified even after implementation. Only the on-site customer can determine if the implementation is according to the stories or not. *Refactoring* is also another practice of XP that directly supports the principle of embracing change. "*Refactoring* is XP's primary means of encouraging programmers to make changes to code that is already working." On simple design practice, programmers implement the simplest design possible and after they learn new information, they perform an ongoing redesign and incorporate the changes over and over again.

Sprint planning meeting in scrum is another practice that welcomes changes. During sprint planning meeting, the different project stakeholders including, customers, users, management, the product owner, and the scrum team decide the next sprint goal and functionality to be done for the coming 30 days and prepare the sprint backlog from the product backlog. The prioritizing the product backlog is handled by the product owner. If there is any change coming, then it will be included in the product backlog and handled appropriately.

- **Identified practices/tools**: XP (*metaphor* and *refactoring*), Scrum (*Sprint planning meeting*).
- **Identified stakeholders**: customers, programmers, users, management, the product owner, testers, requirement analyst, designers, and scrum master.

6.1.3 Simplicity

Simplicity means "to maximize the amount of work not done". Simple things are always easy to use and easy to change when needed comparing with complex things. Fowler and Highsmith (2001) suggested adding things that everyone needed rather than someone needed. Rooney (2011) defined simplicity as: "*we will do what is needed and asked for, but no more. This will maximize the value created for the investment made to date. We will take small simple steps to our goal and mitigate failures as they happen. We will create something we are proud of and maintain it long term for reasonable costs*".

ASD advocates simple design that can solve today's problem instead of complex and general architecture that is assumed to solve future problems. Furthermore simple design makes the next iteration and redesign more encouraging and easier. So ASD is not just about reusability and generality, it is about being simple and focused on current customer requirement and need.

ASD has a motto of simplicity in all the different practices; however XP is the one that directly supports this principle. In XP, its one practice is *simple design* where it is almost similar with this principle. In simple design practice, the two programmers design their code using the simplest way possible using the current requirement available. They are aware of the fact that they may have to do redesign later on but still they will come up with a simple design by not involving anything that is not part of the immediate requirement so this will help them eliminate waste on the way and keep the processes Agile.

The different stakeholders involved include developers, requirement analyst, designers, testers and customers. All of these stakeholders have their own stake on the project. They all have their own responsibility to full fill. In ASD, simplicity is put in to consideration and a guiding principle mainly by the help of the developers. Developers are responsible for making the system less complex, use only the immediate requirement available and simple design to build the system. The immediate requirement is delivered by the customer by the help of the system analyst. And it's the task of the developers to make the design simpler and eliminate unnecessary parts in general try to make it simple just for today so that in long run it's possible to create an environment which the cost of change is low.

- **Identified practices/tools**: XP (*simple design*).
- **Identified stakeholders**: developers, customers.

6.1.4 Customer satisfaction

Agile software development is well known by the principle of putting the customer's need up front. It's not about what the developer or other involved stakeholders want to develop it's what the customer wants. The Agile manifesto incorporates a value called "customer collaboration over contract negotiation". This value underlies unlike other traditional methodologies ASD values the customer's satisfaction more than signing of some legal documents. Customer satisfaction is fulfilled through early and continuous delivery of valuable software. A project may take several months or years to complete so instead of making the customer sit and wait for it that long it is better to deliver working software in smaller iterations. The customer is encouraged to give frequent feedback so that the changes can be incorporated in the next delivery. This will motivate the customer more and even if the project didn't come to completion at the end because of different reasons, the customer still can hold on to a part of working software instead of bunch of plans and documentation. Developing using ASD means creating a new kind of relationship with the customer therefore establishing a healthy relationship and the willingness of the customer will have enormous impact for the success of the project.

The customer here is the key stakeholder involved to have a say on what kind of product he or she wants and on each iteration delivery, the customer closely communicates and works together with few important stakeholders, such as: the main role player that is the developer and project planners and other stakeholders like end users are the ones the service is delivered to. What the developer and the project planner need to consider here is, the system should be according to the customers need and desire (Turk and France, 2005).

ASD has popular practices that have to do directly with customer satisfaction. One of XP's well known practices is *small release,* which is delivering small part of working system that can bring value for the customer. But there are times when the team is unable to deliver something usable but still demonstrate their status and discuss it with the customer if there is a feedback. (Koch, 2005)

- **Identified practices/tools**: XP (*small release*), Scrum (*sprint review*).
- **Identified stakeholders**: customers, programmers, users, project manager, management, the product owner, testers, requirement analyst, designers, and scrum master.

6.1.5 Stakeholder involvement

Stakeholder involvement in ASD is discussed in the principles of Agile manifesto, "Business people and developers work together daily throughout the project" and it is derived from the Agile value of "customer collaboration over contract negotiation". This principle and the Agile value emphasizes all stakeholders should work together daily. This principle divides the project stakeholders in two broad groups; the business people and developers. Developers are members of the technical team which includes programmers, system architects, technical lead, and technical writers and so on.

The business people in ASD are stakeholders that are involved and affected by ASD, which include the entire management (senior and middle management, project manager), supporting services (human resource, finance, contracts, and information technology), the customer (including, their management, technical liaisons, and contracts people), and the end user.

Except FDD all the other Agile practices support the principle of stakeholders collaboration directly in their practices. *On-site customer* in XP is a practice that supports customer collaboration. The customer is the one that can speak on behalf of the end user and he or she answers about what system to be built. XP wants the customer to be available physically with the development team during the entire project life time because of the preference of face-to-face communication. This will give the customer to check if things are going according to plan and so on.

Scrum's *product backlog* is also another practice that directly supports customer collaboration. The backlog consists of all the work to be done by the scrum team and any project stakeholder can add items to the backlog but it is prioritized by the product owner. The product owner collaborates with the project stakeholders in order to decide which functionality to be implemented first and decides if it is implemented according to the item in the backlog.

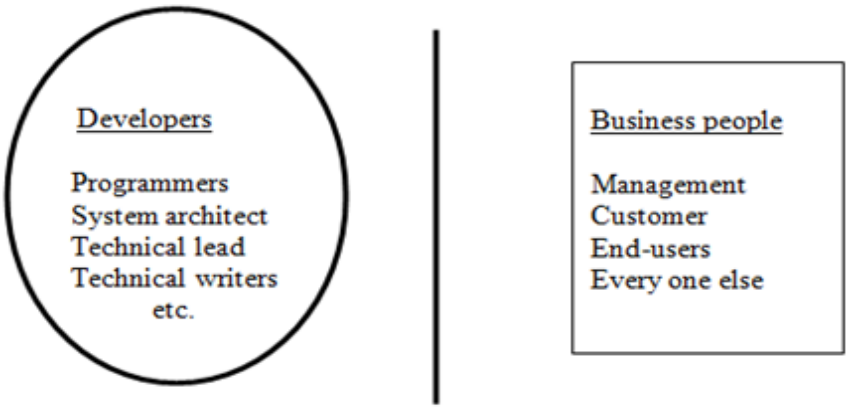

Figure 8: Partitioning of Agile project stakeholders by (Koch, 2005)

- **Identified practices/tools**: XP (*On-site customer*), Scrum (*product backlog*).
- **Identified stakeholders**: programmers, system architects, technical lead/ chief architect, technical writers, management, customers, end-users, product owner, requirement analyst, human resource, finance, contracts.

6.1.6 Motivated and self-organizing teams

This principle is more of a general term used to explain two related principles from ASD. The first one talks about how it is important to build projects under motivated individuals and trust the team to work on their own. The second principle relating teams in agile, talks about the importance of self-organizing team towards bringing best architecture, requirements and design. These two Agile principles about motivated and self-organizing teams, can relate directly with the first Agile manifesto value about "individuals and interactions over processes and tools". ASD highlights the importance of motivated individuals for the success of projects. Moreover creating a suitable environment for the teams and trusting them to work by themselves using the best of abilities they own, can facilitate the road to success. Koch (2005) writes unlike ADD, "the traditional methods assume that intrinsic motivation is rare, and so the enforce specific behaviors". ASD create a suitable working environment with the appropriate technology so that individuals can explore their potential and challenge themselves to con-

tribute. Therefore motivated teams are the outcome of ASD's suitable environment and trusting people.

ASD emphasizes if a team is self-organizing it will lead to a more motivated team. Self-organizing teams posses their own knowledge, perspective, motivation and expertise to apply on the project they are working in. in this kind of environment " the team is treated as a partner with management and the customer, capable of providing insight, affecting decisions, and negotiating commitments" writes (Koch, 2005). ASD sees the concept of self-organizing team as a way to create motivated teams and most importantly to deliver technical excellence. Individuals in a team know about each others capabilities better than any other entity so when they are self organized they can bring the best out of each other and deliver the best technical excellence.

All of the ASD methodologies support the principle of motivated and self-organizing teams on their different practices. *Planning game* and *collective ownership* are two of XP's practices that talk about teams and individuals interactions. *Planning game* in XP is about the team's ability of self-management and motivation. It involves all project stakeholders, including the technical team, management, the customer and so on. In this game every stakeholder is expected to put in his or her expertise to the plate in order to plan the task to be done for the next iteration to be delivered. The customer describes what he or she wants and prioritizes, the team has technical expertise on what to do and how they can do it and the management knows the project's limitation. *Collective ownership* is another practice of XP, which discusses the involvement of teams. XP uses two programmers to work on the same code and these programmers possess a collective ownership of the code they are writing. Both of them can make any update on the code at any time without letting the existing one to fail. Allowed to make this kind of key decision encourages and motivates the team (Koch, 2005).

Scrum's practice called "*scrum teams*" also talks about the Agile principle of motivated individuals and self-organizing teams. In scrum project the scrum teams are the main stakeholders to participate in the sprint planning together with the customer. During the sprint they work on the way they think is right and possible by their own and make decisions and this will motivate them even more.

- **Identified practices/tools**: XP (*Planning game, collective ownership*), Scrum (*scrum teams*).
- **Identified stakeholders**: customers, programmers, users, management, the product owner, testers, requirement analyst, designers, and scrum master.

6.1.7 Sustainable development

Before describing sustainable developments let's see what is happening in conventional project management or software development. We observed that most of the projects start slowly with relaxing mood where it ends up with working long hours even until late night and sometimes in the weekend. These attitudes increase work frustrations and loose connections between stakeholders. ASD follows sustainable development where it means producing a constant flow of product from the beginning to end of the project. ASD never encourages developing a lot of features in a day or in each iteration rather creating a continuous flow of development without any interval (it is like running a marathon rather than 100 meters sprint). Amount of spending time for the whole process should be constant not only for the development team (programmers, designers, testers, application managers) but also for some other stakeholders like sponsors and users. It is project manager's role to establish a sustainable development environment where he/she creates a constant flow of work. On the other hand, the roles for the users are to give direct feedback to the development team in each iteration which ensure better product (Ams technology blog, n.d.).

ASD promote sustainable pace in its processes and the different stakeholders involved in the project, are expected to be consistent on their work. Incremental delivery is one of the mottos of ASD, and each delivery should constitute the same quality as the previous one, should be completed on expected deadline and should establish consistent tempo of work. In order to do that, workers need to be in their normal frame of mind and they shouldn't be forced to work over time. We can't expect to get the same quality of work from worker who gets his or her work done with enthusiasm and full of energy and from one who worked over time and as a result didn't get much sleep, feel tiresome and less enthusiastic.

All ASD practices use incremental development and this make all to maintain sustainable pace on projects however, only XP has a practice that directly supports this principle. *Sustainable pace* in XP is addressed by 40-hour work week. XP considers 40-hour work weeks should be a standard time for employees and don't support the need for overtime. "Frequent over time is rightly considered a symptom of deeper problems, and doesn't lead to happy, creative developers, healthy families, or quality, maintainable code" stated (Larman, 2004). But for some people working only 40 hours a week seems to be abnormal so it is emphasis more on forced over times.

- **Identified practices/tools**: XP (*Sustainable pace*)

- **Identified stakeholders**: programmers, management, testers, requirement analyst, designers, sponsors, users.

6.1.8 Technical excellence

Technical excellence is one of the principles of ASD which enhance agility. Paying critical attention to technical excellence and coming up with good design leads the road to agility. It is may be considered that achieving technical excellence can be costly because it would mean more change on the product, more rework and more retest but ASD considers another way of achieving it. Instead of trying to create technical excellence on the work already done by the programmers, try to use good programming practices that can help create the best design in the first place. Doing that, can make change easy, there will be less rework, retest and in general it makes the review much easier (Koch, 2005). So technical excellence is considered to be the motto of all Agile practices in order to help them be as Agile as they can be.

XP's test first and coding standards directly support technical excellence. XP uses two types of tests, unit testing and functional testing. However while performing unit testing, XP has the principle of writing the test cases first before writing the code and automate the tests. This will force the programmers to think about quality before they do any coding. Moreover, while coding the programmers make sure the code they are writing suits the automated test cases. Because of this continuous testing defects can be eliminated quickly and there will be less rework and in process good design can be achieved. Another practice of XP that enhances technical excellence is coding standards. Establishing coding standards or guidelines are important to maintain the stability and quality of codes in projects. So in a specific project, every team members need to keep in mind to work with common coding standards or conventions. Koch (2005) states, coding standards facilitate other "quality related XP practices" like; pair programming will only work best if both programmers use the same coding conventions, "collective ownership requires that any pair be able to pick up code that is written by any other pair and quickly understands", refactoring works best if the pair understands the existing code and have a confidence to make any change.

Scrum attempts to bring technical excellence to the system using the scrum master as the main weapon. The scrum master is responsible for the project stakeholders, including the technical team, management and the customer are following the scrum processes, tackling any obstacle and making sure the

team is as effective as they can be. Most importantly, the scrum master makes sure if the team delivers high quality. He or she makes sure if the test cases and code reviews are completed, make sure the team is not cutting any quality for the sake of completing the sprint and so on. By making sure the team follows the scrum practices, the scrum master maintains technical excellence in scrum.

- **Identified practices/tools**: XP (*test first, coding standards*), Scrum (*scrum master*).
- **Identified stakeholders**: Programmer, tester, scrum master, designers.

6.1.9 Face-to-face communication

Communication is one of the most important issues in ASD where a better word "effective communication" can be used. In ASD, communication means to transmit information between relevant stakeholders; it could be from developers to customers, from developers to testers, from developers to management stuffs and so on. There are different modes of communications such as face-to-face at whiteboard, face-to-face conversation, video conversation, phone conversation, email conversation, videotape, audiotape or paperwork. Anyone can use any modes for communication but question arise when we talk about effective communication. Statistic shows that face-to-face communication is the most effective way to convey information which has different advantages (Ambler and Lines, 2012). For example when customers and developers work collectively; if any question or issue or problem arises then it can be solve immediately. Absence of face-to-face communication increase miscommunication between stakeholders (Turk and France, 2005).

The most two popular Agile practices (Scrum and XP) have their own principles based on face-to-face communication. Scrum's one of the practices is *daily meeting* which fully satisfy face-to-face communication principle. In daily meeting all of the team members (developers, requirement analyst, designer, architects and testers) should attend. As scrum master and product owner are considered as a team member in Scrum so their presence is also expected in daily meeting. Other stakeholders (departmental VP, salesperson, or developer from other project) may present in the meeting but they are not allowed to talk in the meeting (Cohn, n.d.). On the other hand XP's three practices: pair programming, the planning game and on-site customer are directly related to face-to-face communication principle. In pair program-

ming all technical works like designing, creating test cases, coding, testing, building and integrating system etc are encourage to perform in face-to-face conversation.

In practice sometimes it is really hard to do face-to-face communication, especially in Global Software Development. Kohrell (2011) suggested three ways of communication. First way is of course face-to-face communication as it is the best way of communication where the whole team will locate in the same place. If it is not possible then all team members can meet in a specific time for face-to-face conversation. And finally even if it is not possible then provide them the best tools for the communication.

- **Identified practices/tools**: XP (*pair programming, the planning game and on-site customer*), Scrum (*daily meeting*)
- **Identified stakeholders**: developers, customers, requirement analyst, designer, testers, scrum master, product owner, and project manager.

6.1.10 Product improvement

The last principle of ASD talks about product improvement, "at regular interval, the team talks about how to become more effective, then tunes and adjusts its behavior accordingly". ASD uses project retrospectives as a tool for product improvement in its different practices. "A retrospective is a meeting of all project participants to consider how well the project's processes worked" Stated (Koch, 2005). Retrospective can give insight about what worked best on the project, what didn't work and what are the risks encountered, what to be done next to prevent that problem and so on. ASD emphasizes holding retrospective at regular intervals unlike other traditional methods, which hold retrospective at the end of product delivery. And this has its own down side because after the product is delivered stakeholders are involved with other projects. As a result it will be hard to get everyone together for a retrospective meeting.

Scrum has two types of practices that encourage the team to have a meeting and to solve their problems. The first one is the scrum meeting, which is considered as the backbone of scrum, each week day, same time and same place, with the team members standing in circle answer special questions like; "what have you done since the last scrum?", what will you do between now and the next scrum?" and "what is getting the way of meeting the iteration goals?". The other practice (sprint review) relates to retrospective more

than the scrum meeting. Sprint review is performed at the end of each iteration with the host of the scrum master and other project stakeholders attending, including the product owner. After the demo of the product, the team discusses about the goals, weakness, strength, effort of the team and feedback is encouraged (Larman, 2004).

- **Identified practices/tools**: *retrospective*
- **Identified stakeholders**: developers, requirement analyst, designer, testers, scrum master, project manager.

6.2 Lean principles based on stakeholder perspective

"Lean" is a well-known name in most of the manufacturing companies where Lean has no pure principles. Different authors suggested different principles for "Lean". Although most of the lean principles are related to manufacturing areas, recently many Agile scholars encourage using "Lean" principles in software development. We have derived lean principles that might be useful in software development are discussed below:

6.2.1 Define customer value by eliminating waste

Highsmith (2002) defined value as a collective approach of product features that satisfy customer's needs in a specific time for a specific capital. Womack and Jones (2003) described defining customer value is the starting idea in lean thinking where value is defined by only customers and is created by producers. Meaning of a value is justified only by its specific products. Womack and Jones (2003) suggested lean thinking obviously start with a clear specified value for a specific product including specific price offer for a specific customers. To achieve this Womack and Jones (2003) suggested ignoring existing resources and technologies rather building a strong and dedicated product team; also redefining roles and responsibilities for the team for specific products.

Poppendieck and Poppendieck (2003) suggested eliminating wastes is the best way to achieve customer value where they identified seven wastes are: partially done work, extra processes, extra features, task switching, waiting, motion and defects. Partially doing work is that a programmer might have no idea whether it is going to work or not. A programmer may have well requirements, well designed documents or either may have unit tested code; but he/she cannot say anything until it is integrated. A partially work is never

a part of working software whether it uses resources, time and financial stuffs; which should not be expected. Extra process also considered as a waste. For example paperwork in software development where it absorb resources, make response time slower, hides product quality and which may lost in anytime. So paperwork is not always necessary, yes for safety- critical system sometimes it is necessary. In that case Marry and Tom Poppendieck (2003) suggested making it short, high level and work in off line. Another waste building extra features. A developer may like to add a new feature but behind that feature time, resources are involved. That extra code need to be compiling, testing and integrating which may increase complexity and potential failure. Task switching makes working progress slower where Agile's another principle is delivering as fast as possible. In software development delays could arise from while starting a project, involving employees, excessive requirements documentation, reviewing and approvals, testing etc. Delays are natural but decreasing delays increase customer values and satisfaction. Motion is considered as another waste where it is necessary to keep together all of the stakeholders like requirement analyst, system designer, programmer, tester and customers. Defects are also considered as a waste where continuous integration and testing can minimize defect problem.

Middleton, Flaxel and Cookson (2005) suggested achieving a successful requirement analysis stage to ensure customer value. All members of cross-functional team are encourage to attend in requirement analysis part which will help them to come up with a common prioritize list of features. Middleton et al. (2005) also suggested identifying the most important people from customer side which will help them to deliver product earlier. Even a successful requirement analysis minimizes rework. Power (2010) includes developers, testers, designers, architects and document writers in cross-functional team. For a successful project it is necessary to read customer's from the beginning of the project, also needed to identify customer's want.

The basis of LSD is to eliminate waste where one of its practices JIT has been used directly to eliminate waste. JIT helps the team to do the right things in the right time while considering right amount. Another strong practice is value stream mapping where Poppendieck and Poppendieck (2003) mentioned value stream mapping as a tool and Womack and Jones (2003) mentioned it as a principle. If we consider value steam mapping as a function then we identified this function has two parameters: task and duration. Value stream mapping encourage a manager to go through from the beginning of the project to end while traveling each task and duration to accomplish corresponding task. This practice will help the manager to come up with a structure of the overall development with identifying unnecessary work which we are considering as a waste.

There is also another kind of waste to consider eliminating in LSD, which has to do with people downtime. It is considered as behavioral waste which arises due to poor interpersonal relationships or lack of communication between workers. This kind of problem most of the time is tolerated by management and given little emphasis. Emiliani (1998) mentions how difficult it is to eliminate behavioral waste because it is hard to measure unlike production waste. But we know for a fact that "the persistently wasteful individual and group behaviors could be a reason why many large businesses fall well below the expectations of one or more of its stakeholders"(Emiliani,1998).

The possible stakeholders could be customer, developers, testers, designers and document writers. Specifying value for the product is all about the customer's need and expectation. It is not an easy task to view value through the customers' eyes, it may means to consider the best way of communicating with them, understand what and how they want the employee to be and have the best of interpersonal relationship while interacting with them. Employees may have hard time following this principle because of feeling they know best than the customer and want to do things in their way. So to undergo this process most companies need to work on their people's mindset, business processes and so forth (Emiliani, 1998).

- **Identified practices/tools**: *seeing waste, value stream mapping, JIT.*
- **Identified stakeholders**: customer, developers, testers, requirement analyst, designers, document writers, management, and project manager.

6.2.2 Decide as late as possible

Let's think about breadth-first search (BFS) and depth-first search (DFS) for a given tree. BFS explores each node at a given depth before moving to the next level where in DFS traverses as deep as possible before traversing another higher level child. In sequential software development all the requirements are gathered first and then designers are forced to apply depth-first decision to go for a solution. While taking depth-first decision it discovers low-level solutions where high-level solutions could be present. Depth-first decision encourages the designer to identify a solution and solve it as soon as possible. Applying depth-first decision is the most costly mistakes as requirements may change in anytime by customers. ASD/LSD encourages concurrent approach where iterative development has been used. Concurrent approach helps the designer to take breadth-first decision where best product can be developed based on current requirements. Breadth-first decision helps

the designer to traverse all the possible solutions before taking any decision which is more cost effective, reduce delivery time and improve product performance at the end. It is possible to reduce cost incredibly by taking decision as lately as possible. Statistic mentioned that finding and fixing software problem in early design phases is 100 times less than finding and fixing problem after delivery (Poppendieck and Poppendieck, 2003 cited in Barry Boehm, 1987).

There are different changes where few changes are very important and need to decide at the beginning of the project for example choice of languages, architectural layering decision or the choice of database that is going to interact with other application. As Marry and Tom Poppendieck (2003) suggested taking breadth-first approach in such decision as changing decision rise cost 100 times more.

Poppendieck and Poppendieck (2003) discussed about three tools (option thinking, the last responsible movement and making decision) to satisfy this principle. Option thinking is to find a way to provide opportunity for the customers to take decision delay with ensuring the same benefit. Concurrent software development helps to take decision as late as possible, but still there may such situation arise where a good decision may not come before last responsible movement and then decisions are taken by default. In such case different policy can be adapted like: share partially complete design information, direct worker-to-worker collaboration, creates knowledge (how to absorb changes, what is critically important in the domain, when decision need to make and quick response capability). While making decision two types of decision can be consumed either breadth-first decision or depth-first decision. When new problem arises and need to solve it quickly then breadth-first approach is much better in such situation. To use the benefit of depth-first decision in software development two things are required: an expert who can take decision early without any mistake and have to ensure that no requirement will change once it decided. On the other hand breadth-first decision is best for unstable domain where requirement is not fixed.

To satisfy concurrent approach Poppendieck and Poppendieck (2003) suggested that developers should be experts in their specific domain where they will work closely with customers and designers. Still there are few requirements that need to change all the time during the whole product development, there is no way to build such a general product or feature that will satisfy customer's requirements all the time. In such situation Poppendiecks suggested to build a simple solution rather than making it complex; it will help designer or developer to change faster when needed.

- **Identified practices/tools**: *option thinking, the last responsible movement and making decision.*
- **Identified stakeholders**: customers, programmers, project manager, requirement analyst, designers, and testers.

6.2.3 Move towards flow and deliver values fast

In software development customers are all in all where delivering product as fast as possible is one way to satisfy customers. Rapid delivery is an operational practice where it has two significances. One is it allows some customers to take their decision as late as possible and on the other hand it satisfy other customers by delivering fastest product. Poppendieck and Poppendieck (2003) mentioned "rapid delivery often translates to increase business flexibility". Rapid delivery allows the company to deliver faster by using fewer resources so that customers can change their minds which can also reduce risk. For example in software development when developers write a lots of code without testing, or may be code is well-tested but incomplete integration; it creates risk. Such risk can be dramatically reduced by delivering product fast. To deliver fast product Middleton, Flaxel and Cookson (2005) also suggested to work with small batches for analyzing requirements, designing, coding and testing can be done earlier which can also detect errors in earlier stage.

To deliver faster product, it is considered to be a great idea to follow lean flow principle. When values are identified from customers then these values are divided and mapped based on customer's needs which is known as value stream mapping. While developing these values it is necessary to create a flow which ensure products are delivering sequentially without any delay (Womack and Jones, 2003; Julien and Tjahjono, 2009).

Rather than thinking to developing software, teams should consider it as a flow of values where small changing is adapting in each iteration. To make values flow perfectly Middleton, Flaxel and Cookson (2005) suggested taking care on load sharing where it ensure that tasks are assigned on the team based on their skill and expertise.

Flow ensures a continuous flow of values to the customers without any delay. It is the basic element of continuous improvement because to create a flow of development it is necessary to reduce batch size, cycle time, delay, WIP and other wastes (Larman and Vodde, 2009).

Poppendieck and Poppendieck (2003) mentioned three tools for faster delivery: pull system, queuing theory and cost of delay. In pull system it encourage that team should pull the work rather than pushing work by project manager. To achieve pull system team should be self-motivated and self-directed. In LSD customers use the advantage of pull system where they pull work for the development team rather than a schedule push work. Pull system also increase team's motivation as task are defined by customers and fulfilled by team where the absence of project manager. Kanban is the best practices for adapting pull system in LSD. Another way to deliver product as fast as possible is by adopting queuing theory in the production system where the main characteristic of queuing theory is cycle time (queue time + service time), rate of arrivals, rate of services, utilization and bottleneck. And the last tool is calculating cost of delay. A brief description is described in "Developing Products in Half the Time" by Preston Smith and Donald Reinertsen.

- **Identified practices/tools**: *pull system (Kanban), queuing theory and cost of delay.*
- **Identified stakeholders**: customers, programmers, project manager, requirement analyst, testers, designers, finance/ accountant.

6.2.4 Continuous improvement

According to Larman et al. from the Toyota way, (there are two pillars of lean: continuous improvement and respect for the people. Continuous improvement can be achieved by involving everyone but most importantly "the true value of continuous improvement is in creating an atmosphere of continuous learning and an environment that not only accepts, but actually embraces change". However, none of this will make sense if there is no respect for people. Ongoing improvement targeting perfection as the only goal is considered to be one of the main principles of LSD. According to Larman and Vodde (2009) the success of continuous improvement in LSD depends on four elements: *go see for yourself* (instead of sitting in the office and receiving information, LSD encourages for the different stakeholders, specially managers to go to the place in person so that they can understand the situation better and in process this can contribute for improvement), *kaizen*(Continuous improvement often called as kaizen, which is a Japanese word referring to the constant strive to perfection (Karlsson and Åhlström, 1996)), *perfection challenge*(to have a higher expectation and to challenge every stakeholder involved for better product), and *work toward flow*(while

working toward flow, there will be more weaknesses and wastes found and this will provide opportunities to continuous improvement).

Moreover, standardization is another key element of continuous improvement (Ohno, 1988; Liker and Morgan, 2006). While developing product it is necessary to reduce variation and inventory. Standardization helps to reduce variation, remove wastes and allow developing high quality product. It also has been observed that when different people perform the same work in different time, the quality of product vary from person to person if there is no standard procedure. There are different ways to increase standardization where Calderone (2010) describes role of clarification, performance standards, drawing workflow, checklist etc could be a good way. Role of clarification is that all the employees are aware about his/her working area, performance standards refers there should be a standard of measuring performance, drawing workflow helps the employees to understand how works flows through different process and what happen if one task is not completed, finally checklist includes key process steps that must be perform while doing a particular task.

Kaizen is a well known practice of LSD that is directly related to continuous improvement. It is a process of allowing the team to come up with the best technique and practice to do things and to experiment these technique further until they find a better way and repeat this over and over again. Kaizen involves a widely used tool called five whys (5 whys), which means "In response to a problem or defect, a team considers "why?" at least five times". The LSD team can apply the 5 whys to identify problems from the source while stopping and fixing problems.

- **Identified practices/tools**: *go see for your self, kaizen, perfection challenge, work toward flow, standardization.*
- **Identified stakeholders**: managers (including senior managers), programmers, requirement analysts, testers, designers.

6.2.5 Create learning environment

Creating learning environment is one of the most popular principles in LSD. Although the origin of lean comes from production system but here we are using it in development system. Propendieck and Propendieck (2003) mentioned a clear difference between development and production. In development product quality should fit for use where variable results make it better

and values are generated by iteration. And in production product quality confirms that it satisfied all the requirements where variable result creates poor product and waste is generated by iteration. Learning can be achieved from different stage of software development. For example in design phase normally designers use top-down approach when problem is well-defined. But what happen when problems are not well-defined? Rather than taking top-down approach designers use repeated scenario examination, requirements elucidation, high-level solution segmentation and low-level design of difficult elements (Marry and Tom Propendieck, 2003) which is called learning cycles. The same situation may arise for requirement analysts, architects, programmers and testers.

Propendiecks also suggested involving business people to amplify learning where delivering working software also creates knowledge. They discussed four tools for amplifying learning as feedback, iteration, synchronization and set-based development. Feedback is one of the strong tools for amplifying learning. In traditional software development feedback considered as a threat because such learning involved changing pre-determined plan. But in today's LSD/ASD, iterative development and feedback are necessary where teams and customers work simultaneously. In each iteration, features are selected based on customer's importance where features also need to synchronize after each change. Propendieck and Propendieck (2003) suggested three ways (synch and stabilize, spanning application and matrix) to synchronize features. Synchronize daily in a single team but if there are multiple teams, synchronization is performed in a weekly matter. Moreover, to get a rapid feedback, its advised to do frequent builds. And finally we use set-base development where key features are: develop multiple options, communication constraints and let solution emerge. Koch (2005) writes set-base development is a unique approach to identify technical related problems. it encourages the entire development team to identify the most excellent solution to problems encountered and results to a higher quality system

LSD encourages and try to make everything a learning experience and emphasizes knowledge transfer moreover encourages the team to respect and share the knowledge and challenge other stakeholders like partners in to lean thinking and help them improve.

- **Identified practices/tools**: *feedback, iteration, synchronization and set-based development*.
- **Identified stakeholders**: programmers, requirement analysts, testers, designers, and partners.

6.2.6 People focus

In this principle authors indicate two important areas of an organization that are: empowering team and organizational structure. Empowering team is one of the key principles in LSD. Traditionally the overall project controlled by project managers where traditional project manager's roles can be defined as directing the overall work, take decision to fix a problem, hire and fire employees, knows the goal, identify and control fault, knows the solution and pride in personnel's success. On the other hand lean project managers work different and far away from traditional project manager's concept where roles and responsibilities can be summarized as: assigning process responsibilities to the team, developing problem solving teams, encourage team for better product, provide all necessary resources, take solutions form team and pride on team's success (Miller, 2005).

For empowering the team Poppendieck and Propendieck (2003) suggested four tools: self-determination, motivation, leadership, expertise. Self-determination is let the team what they want to do as team already trained how to design its own work procedures. Management's roles are here to coach the team and provide all necessary resources when needed. It helps the team to come up with new ideas or proposal for better production where management verify those ideas and take decision based on those ideas. It is necessary to ensure motivated employees in a team because a motivated team does not need to do rather the team identify what to do. It is also necessary to create intrinsic motivation to the employees which requires feeling of belonging, feeling of safety, sense of competence and sense of progress. Leadership is one of the key tools for empowering team where a leader's roles could be direction setting, align people based on their expertise and enable motivation. The necessity of leadership in a team is also discussed in (Larman & Vodde, 2009; Liker and Morgan, 2006; Åhlström and Karlsson, 1996, 1998). And finally it requires involvement of expert in the team for better production.

Besides focusing on team, LSD also encourages to involve other stakeholders who are more or less important (for example operations, support, financiers etc). Rather than considering as an "IT" department Poppendieck and Cusumano (2012) suggested considering as a product development department.

To flow information, decision and resources efficiently it is necessary to have a proper structure in the organization. Most of the authors suggested for cross functional team (matrix structure) where different experts are involved in a team (Ohno, 1988; Åhlström and Karlsson 1996, 1998). A cross func-

tional team increase direct feedback and corrective action by reducing unnecessary delay. In software development experts could be database experts, user interface experts, embedded code experts, middleware experts and domain experts Poppendieck and Propendieck (2003).

- **Identified practices/tools**: *self-determination, motivation, leadership, expertise.*
- **Identified stakeholders**: project manager, designer, programmer, tester, domain expert, management, customers, and financiers.

6.2.7 Customer focus

A business runs based on customer's loyalty. Success of a business builds on customer's satisfaction. Professionals believe that satisfied customers may bring new customers in, where dissatisfied customers may encourage not going for service from that business. A customer has plenty of choice for a specific service, but when he/she choose a particular business for that service; business manager should understand that situation and provide the best solution to satisfy the customer. Special training and guidelines are introduced in lean six sigma management which is very popular in lean manufacturing where that knowledge can be introduced in LSD. It will help employees to deal with several customers in the same time where each customer will feel special by themselves. It is also suggested to use the same supplier again and again to deal with the same customers because customers have no time to introduce themselves and create trust on different suppliers.

Now question arise how we can achieve customer's satisfaction? And the easy answer is just fulfilling customer's needs. Professor Noriaki Kano developed a model in 1980's known as Kano model which talked about three types of customer's needs: basic, performance and excitement. Basic needs are those needs which are fulfilled by the product or producer as customer expected. Basic needs are so general that failure of basic needs directly dissatisfy customers. That's why it is necessary to fix basic needs failure immediately when it arises. For example if we think about software development basic needs could be *develop bugs free product, no spelling mistake in the product, works efficiently and of course on time delivery* etc. Performance needs have a proportional impact on customer satisfaction. If we think about cost in software development; the more cost will be increase the more customer satisfaction will be decrease. So while dealing with customer's managers should consider about performance needs (cost, lead times, purity etc) which have an impact on customer satisfaction. Excitement needs are those

needs where customers have no idea. Failure of excitement needs have no impact on customer satisfaction but involving a successful excitement needs have a great impact on customer's satisfaction (Miskelly, 2009).

When we are thinking about satisfying the customers; we also need to think about the communication media as it is one of the most important issues to satisfy customers. They are different way of communication like face-to-face, phone, email etc. But for effective communication Highsmith (2002) suggested face-to-face communication. Face-to-face communication involves customers on site which is the key for developing successful product. To satisfy customers it is always necessary to give the first priority on customer's needs. To achieve this lean's one of the most popular practice Kanban can be use where it will helps the team to develop exactly the right product when it is needed.

- **Identified practices/tools**: *Kanban, feedback, option thinking, queuing theory.*
- **Identified stakeholders**: customer, developer, designers, testers, requirement analysts, management.

6.2.8 Product excellence

Product excellence is one of the key principles in LSD where it refers how to come up with good quality software, how to improve software quality, how to satisfy market demand. Different authors explain different principles to achieve product excellence. Poppendieck and Propendieck (2003) suggested "build integrity in" where they describe product integrity has two scopes: perceived integrity and conceptual integrity. "Perceived integrity means that the totality of the product achieves a balance of function, usability, reliability, and economy that delights customers. Conceptual integrity means that the system's central concepts work together as a smooth, cohesive whole". Relevant stakeholders are also identified from both areas where in perceived integrity it includes domain experts, users, customers and product owner and conceptual integrity includes developers, master developer, operators, maintainers, analyst and testers are involved. In perceived integrity it ensures flow of information flow from customers/user to team where in conceptual integrity it ensures technical information flow between team members. Different models (conceptual domain model, glossary, use case model, qualifiers) that can be useful for successful information flow while developing complex system. While achieving conceptual integrity Poppendiecks (2003) suggested four guidelines for complex designing: use possible existing solu-

tions, use integrated problem solving, use experienced developers, and use master developer. Refactoring is also another key of continuous improvement. And finally the authors mentioned testing (customer testing) is another tool for product excellence where it runs throughout the development. Another way to improve product quality is to create an environment for visual quality measurement which can measure quality automatically (Middleton, 2001).

Product excellence can be facilitated using practices like linked processes (linking the processes with their component by putting them together), which can mean trying to keep projects in the same building so that people don't have to travel here and then to do their work and standardized procedures (can help people to be moved from one projects to another easily). This practices increase productivity and speeds product delivery stated (Middleton, Flaxel & Cookson, 2005).

It is also possible to borrow some other lean tools for product excellence that has been already used for other lean principles. Such one tool is set-base development where all the possible solutions bring first and then analyze each solution. And finally the best solution is selected and implemented based on relevant information. Another two tools leadership and expertise directly effect on product excellence. Leader creates a vision for the developed product and helps the team to achieve that vision, also encourage loyalty for the product and motivate for good teamwork. Besides leadership it is also necessary that team members should expert on their area of work which also provides different solution in brainstorming (Koch, 2005).

To achieve product excellence it is also necessary to adapt technologies. No doubt about using technologies for better product development; however we have to make sure we are using the right technology suited for the need of the company. Efficiency can be brought to products through a hard working team but in order to make the team even more productive we can bring technology as an extra tool. Now a day's technology changes every time, therefore it's the manager's responsibility to identify what kind of technology needed for them and help the employees to adapt to it (" need determine technology"). Ohno (1988) encouraged using automation while developing product; it can be advantageous in control over production (reduce waste) and prevents producing defective products. In software development automation could be introduced for detecting errors while testing. If we think of a team; adding automation increase the skill of each team member's efficiency. Liker and Morgan, (2006) also encouraged to use technologies that fit with employee's need. "Visual control can bring production weakness to the surface" Ohno (1988). For information visualization lean tool Kanban and Hoshin Kanri can be more useful.

Toyota emphasizes the use of simple visual tools like big colorful physical cards on the wall so that people can touch and move them when they want. These tools can help the team point out problems, communicate and coordinate the pull system easily, stated (Larman and Vodde, 2009). These kind of visual tools are better than displaying small detailed information on some kind of software.

- **Identified practices/tools**: *perceived integrity, conceptual integrity, customer testing, linked processes, set-base development, leadership, expertise, automation and visual control.*
- **Identified stakeholders**: domain experts, users, customers, developers, master developer, operators, maintainers, analyst and testers.

The following table summary shows how we have derived lean principles based on different author's references:

Table 2: Derived lean principles based on different authors.

Principles	Description	Reference
Process		
Define customer value by eliminating waste	Value defined by customer; always provide the best value for the money; by preventing defects, eliminating rework, minimizing unnecessary documentation waste can be eliminated.	P1, BM1, WJ1, M3, MEA2, MEA7, H2, H9, HO1, ÅK1, PC2, LM1, E2, O1.
Decide as late as possible	Accept changes in any time. Explore all alternative solutions from root cause to achieve the best solution.	H5, H6, P3, BM4, LV12, LM2, O6, O11.
Move towards flow and deliver values fast	Establish continuous flow of processing to deliver fast; an 80% today than a 100% solution tomorrow; Create an environment to flow customer's values. Creating level for process flow. "Smaller work package sizes, smaller queue sizes and reduction in variability".	P4, BM5, MEA1, H8, PC5, WJ3, MEA3, LV2, HO2, LM3, O5, O9.
Continuous improvement	Continual quality improvement, project after project. Achieve standardization for product development that reduce inventory, adapt stopping and fixing problems.	M1, MEA6, MEA11, LV4, LV5, HO5, HO6, ÅK2, LM4, E5, LV13, O4, O10.
People		
Create learning environment	Create knowledge from each iteration. Help partners be lean.	P2, BM3, LV11, PC4, LM10
People focus	Respect people. Empower the team. Develop exceptional people. Multifunctional team and decentralized responsibilities. Create leader. Balance functional expertise and cross-functional integration, vertical information system, base management decision on long-term philosophy. Engage everyone.	P7, BM6, M2, H4, H7, LV6, LV10, HO3, PC6, LM7, LM8, LM9, E3, ÅK5, ÅK6, LV9, ÅK7, LM5, LM6, MEA8, ÅK8, LV1, O7, O8.
Customer focus	Customer satisfaction, active customer participation and establish pull scheduling.	WJ4, H1, H3, LV3, ÅK4, E1, O12.
Tools		
Product excellence	Continuous integration of small unit, striving for excellence, well-tested technology, Automatic quality measurement devices and understanding lean. Adapting technology to fit employees, use powerful tools for organization learning, needs determine technology; introduce autonomation to make employees more effective.	P5, BM2, WJ5, M4, M5, P6, BM7, PC1, WJ2, MEA5, MEA4, O2, MEA9, H10, H11, LV7, LV14, HO4, LM11, LM12, LM13, E4, O3.

6.3 Summary

In this chapter we have derived Agile and Lean principles for software development. While deriving Agile principles we have followed Agile Manifesto principles as these principles are using most of the software development companies. Besides we have identified relevant practices and stakeholders under each principle. As Lean has no pure principles for software development we have derived Lean principles based on different author's suggestion (see section 5.2 for details). While deriving Lean principles we have chosen only those principles that are closely related to software development where relevant practices and stakeholders are also identified. The contribution of this chapter is if anyone wishes to use Agile or Lean principles separately they are welcome to follow these principles where relevant practices and stakeholders are also identified. The list of Agile and Lean principles are summarized below:

Agile Principles:

1. Working software frequently
2. Embrace change
3. Simplicity
4. Customer satisfaction
5. Stakeholder involvement
6. Motivated and self-organizing teams
7. Sustainable development
8. Technical excellence
9. Face-to-face communication
10. Product improvement

Lean Principles:

1. Define customer value by eliminating waste
2. Decide as late as possible
3. Move towards flow and deliver values fast
4. Continuous improvement
5. Create learning environment
6. People focus
7. Customer focus
8. Product excellence

7. Analysis

This analysis chapter is the heart of our thesis. In this chapter we will identify the difference between Agile and Lean based on stakeholder perspective. To identify the difference between Agile and Lean, we are going to use section 6.1 and 6.2 where have already derived Agile and Lean principles. Moreover, we have also identified different stakeholders from each Agile/Lean principles. To show the importance between stakeholders we are going to use Mitchell's model.

7.1 Comparison work

In this section we are going to analyze the difference between Agile and Lean principles based on stakeholder perspective. Moreover, we will also show the relationship between stakeholder theory and Agile/Lean stakeholders.

7.1.1 Agile principles Vs Lean principles

In this section we are going to compare our derived Agile and Lean principles. While comparing Agile and Lean principles we have chosen the most nearest principles from both sides where the principles are almost same on their objectives or scopes or aims but could be different the way of thinking and involving stakeholders. To achieve this comparison we have used Weber "Ideal type" concept. According to Weber, an ideal type is a useful tool for comparing and analyzing social or economic phenomena. We reconstruct the ideas, based on established well-cited literature on two ideal types "agile" and "lean" in order to compare conceptually-analytically. For example working software frequently (see section 6.1.1) is one of the agile principles and move towards flow and deliver value fast (see section 6.2.3) is one of the lean principles. The objectives of these two principles are almost same but the ways of achieving these objectives are different. From agile section small releases, continuous integration, sprint, sprint review etc could be a good way for delivering software fast. On the other hand different lean scholars suggested that rather than considering software development, think software development as a sum of values where values are flows on

different interval based on customer's needs. So in our comparison part we are going to compare these two principles. Similarly embrace change (see section 6.1.2) is one of the agile principles where any change of requirement is always welcomed in agile. On the other hand Lean has almost the same principle known as decide as late as possible (see section 6.2.2) and continuous improvement (see section 6.2.4) which are not only welcome late requirement changes but also suggest how to minimize late requirement changing requests. So we will compare embrace changes versus decide as late as possible and continuous improvement. Let's see the following comparison relationship diagram before analyzing their relationship.

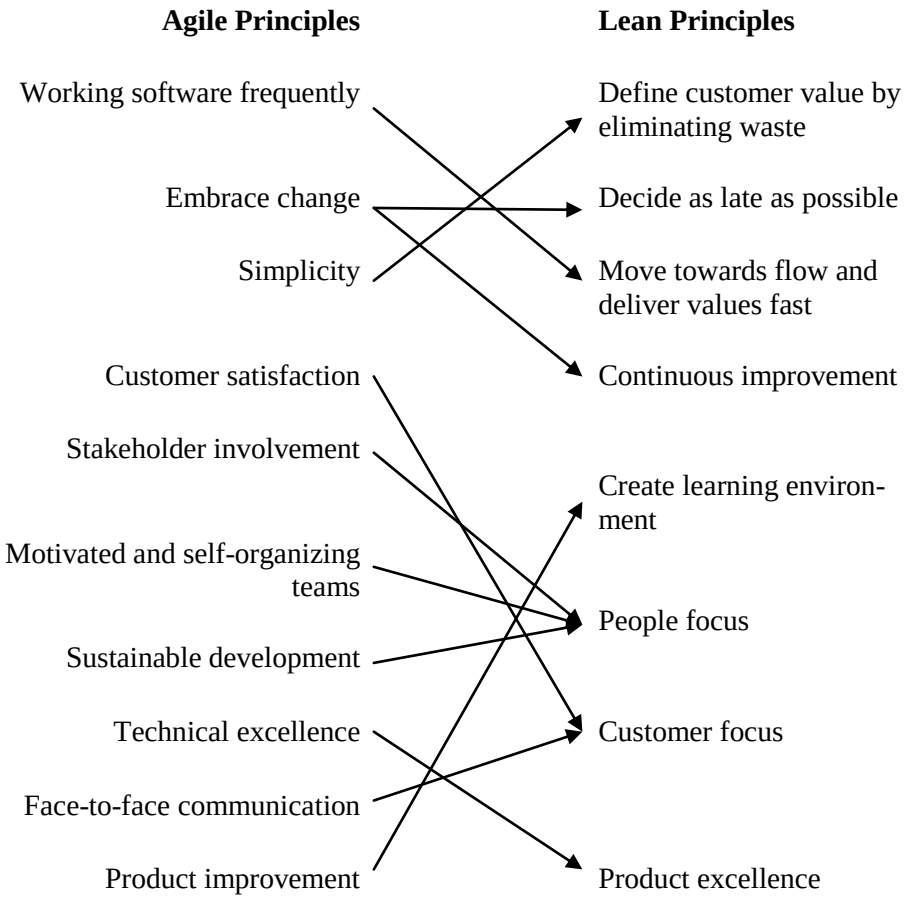

Figure 9: Possible nearest Agile versus Lean principles

7.1.1.1 Working software frequently Vs Move towards flow and deliver values fast

Delivering working software frequently is the main key in ASD as the correct delivery of product creates values to the customer. Progresses of works are only measured by working software. To achieve this principle several practices can be involved like from XP (small releases and continuous integration) and Scrum (sprint and sprint review). Obviously small releases and continuous integration help the team to come up with working software frequently. Sprint and sprint review also provide common decision, plan for next iteration and one way of learning from the previous sprint. And to bring out these principles we identified programmers, customers, requirement analyst, designers, testers, scrum master, product owner, management (senior manager, middle-management, project manager), and users are the key stakeholders in ASD.

On the other hand LSD has the same principle called "Deliver flow of values as fast as possible". Rather than considering product LSD suggested it as a sum of values where values could be flow smoothly in different iterations. In ASD for delivering software frequently different group meetings (sprint, sprint review) held on where in LSD we did not identify such group meetings where all the key stakeholders are present. Rather we identify three tools: pull system, queuing theory and cost of delay that could be useful for frequent product delivery. Kanban is a well-known tool that satisfies pull system where orders are taken directly from customers. Queuing theory helps to deliver product fast with minimum delay. Besides delivering product fast, LSD also suggested to calculate cost. Cost of delay provide a clear cost calculation that a manager or financier can calculate. In LSD we identify programmers, customers, requirement analyst, designers, testers, manager and financiers are the key stakeholders.

7.1.1.2 Embrace change Vs Continuous improvement & decide as late as possible

The second principle in ASD is embrace change where new requirements are always welcome. This is one of the ways to satisfy customer. To achieve this principle XP uses two practices (*metaphor* and *refactoring*) where Scrum uses *Sprint planning meeting.* Refactoring create a good relationship and respect between programmer where *Sprint planning meeting* involves all key stakeholders. In *Sprint planning meeting* customers are welcome for changing any requirements. To satisfy this principle using these practices we identified customers, programmers, users, management, the product owner, test-

ers, requirement analyst, designers and scrum master are the main stakeholders.

On the other hand in LSD, we identify two principles decide as late as possible & continuous improvement which are mostly close with embrace change principle. In LSD, changing requirements are also welcome but LSD also deals with how to acquire better requirements from the beginning of the project that can reduce changing requirements requests on next iteration. Decide as late as possible principle can be use to achieve better requirements from the beginning of the project which has three strong tools (option thinking, the last responsible movement and making decision), though it is better to avoid last responsible movement. The key stakeholders under this principle are: customers, programmers, project manager, requirement analyst, designers, and testers.

Acquiring better requirements from the beginning of the project not only can reduce changing requirement request but also need proper development work in each iteration. Enabling continuous improvement can perform vital task for proper development. Continuous improvement has also four tools (*go see for yourself, kaizen, perfection challenge, work toward flow, standardization.*) that can be useful for product improvement. It is also necessary to follow a standard procedure called "standardization" which can also reduce rework. In this principle we identify manager, programmers, requirement analysts, testers, designers are the key stakeholders.

7.1.1.3 Simplicity Vs Define customer value by eliminating waste

Simplicity is one of the key principles is ASD where it concern on to do this things in a simplest way. As ASD is a dynamic procedure therefore satisfying simplicity principle helps the developers to synchronize their work easily when changes are required. One of the XP's principle *simple design* talk about simplicity directly where it encourage that two programmers should design their code in a simplest way using available requirements. We identify developers and customers are the key stakeholders for achieving simplicity principle.

On the other hand LSD is not too far away from simplicity. Removing wastes are the one way to achieve simplicity where LSD described seven sources of waste briefly. Considering those wastes gives boarder knowledge to the manager and team; also helps them to perform their own work more systematically. Another LSD tool JIT ensures that the team doing the exact required things which are another way to remove waste. Value stream map-

ping helps the manager to come up with the overall development structure by identifying each process with corresponding durations. While comparing simplicity in ASD with LSD we identify ASD talk about developers how to come up with a simple design where in LSD it talk about in a general way to achieve simplicity. But we belief that simplicity is not only for the developer, rather than achieving simplicity, it is also required to practice for other stakeholders like requirement analysts, designers, testers and could be for project managers where everyone should try to come up with their own work in a simplest way. The required stakeholders for simplicity in LSD are: customer, developers, testers, requirement analyst, designers, document writers, management, and financiers.

7.1.1.4 People focus VS Stakeholder involvement; Motivated & Self organizing Teams

People focus in LSD is all about the different stakeholders (internal and external stakeholders) that are involved in the project. LSD gives as much as freedom to the stakeholders as ASD does, unlike other traditional project management tools LSD gives emphasis to things like; respecting people, empowering the team, developing exceptional people, multifunctional team and decentralized responsibilities, creating leader, balance functional expertise and cross-functional integration, vertical information system, through management decision on long-term philosophy and so on. With some similarities ASD have a philosophy of all the stakeholders working together in a daily basis and has a strong believe on motivated and self-organizing teams. Both ASD and LSD's success depends on their team so both agree on creating a suitable environment for the team and believing the teams to work on their own. However, LSD doesn't consider need of self-organizing teams like ASD does, because informal team organization is considered as disastrous. Both consider face-to-face communication as a major way of communicating with the team members and also other stakeholders involved, specially the customer. The difference we consider here is that LSD gives a great deal of emphasis and considers it as an important tool to create good leaders. The role of leadership is appreciated in LSD but given lesser emphasis in ASD. LSD uses paternalistic management system while ASD follows laissez faire management that encourages the team to openly share information through virtual technology. In stakeholder wise, both seem to involve similar type of stakeholders, except LSD involves the end user less than ASD. And ASD involves some special stakeholders like scrum masters and end users. ASD concentrates on the development team more than the entire stakeholders, for example the management; on the contrary LSD is a management tool that involves all project stakeholders including the manage-

ment and other external stakeholders. LSD considers everybody in sight being a stakeholder: employees, managers, every suppliers, and partner) whereas, the agile manifesto emphasizes customers. However ASD gives precedence to people over processes and tools, while LSD focuses on processes than people.

7.1.1.5 People focus VS Sustainable development

ASD advocates how the different processes and stakeholders (the sponsors, developers and users) should keep sustainable pace. Because of their practice of incremental delivery ASD makes sure the team delivers consistent quality of work to the customer. ASD don't support the need of over time because it adds extra burden to the workers and has a negative impact on their performance. Where as, LSD advocates mass production and continuous improvement, while following this principle LSD gives excessive emphasis on process discipline and frequent rotation of workers and these kind of practices can pressurize workers and force them to lack motivation on their job. Unlike ASD, in LSD working frequent over time is not considered as a big problem that can prevent the team from maintaining a sustainable pace in their work. Instead, LSD emphasizes maximum efficiency from individuals or team members performing a single task.

7.1.1.6 Customer focus VS Customer satisfaction; Face-to-face communication

Both ASD and LSD give highest priority to customer satisfaction. ASD achieves customer's satisfaction through early and continuous delivery of valuable software. In each delivery, the customer can give feedback so that it could be incorporated in the next iteration. ASD is well known by its way to adapt to requirement change that comes from the customer. These can be achieved by the customer being available face-to-face with the development team. But close communication of the customer is with the developer and with the requirement analysts than the other project stakeholders. ASD emphasizes delivering valuable software to the customer rather than concentrating on signing legal documents. With some exceptions, it is the same for LSD, delivering what the customer needs instead of what the development team can be able to deliver. LSD also tolerates change when it comes to the customer but not as ASD does. LSD also prefers face-to-face communication with the customer but it may not be as often as ASD does. LSD has a practice called "pulling from the customer's back", which has to do with only manufacturing when the customer orders it and its well known by mass pro-

duction. Unlike LSD, ASD's goal is to come out of mass production and when it comes to the customer ASD can respond to change, handle unpredictability and uncertainty in the business environment better than LSD. In conclusion LSD calls it customer defined value while ASD prefer customer collaboration; both empower the customer to choose what he or she wants.

7.1.1.7 Create learning environment VS Product improvement

Creating learning environment has been one of the principles of LSD. Learning can be created in each iteration not only for the team but LSD encourages the different stakeholders like customer and partners to get the knowledge. LSD uses different tools to amplify learning: feedback, iteration, synchronization and set-based development. So LSD directly supports creating learning environment in its principle and contains important tools to help do that. On the other side in ASD, even if we don't find written principle under its name, some of the principles encourage creating learning environment, such as product improvement. There is a commonly known tool under this i.e. retrospective; it helps the team to sit and discuses the problems, advantages, risks and so on, therefore this will help the team to transfer knowledge.

7.1.1.8 Technical excellence Vs product excellence

Technical excellence is one of the key principles in ASD where it talk about coming up with good design. Two XP's practices test first and coding standards directly support this principle. These practices help the programmer and testers to perform their work more efficiently and as well as improve the quality of the product. In scrum, scrum master itself is considered as a tool where scrum master is responsible for providing all necessary support for the team. In ASD, we identified programmers, testers, scrum master, management, and customers are the relevant stakeholders for technical excellence.

In LSD we identified product excellence principle which can be compared with technical excellence from ASD. For product excellence different authors suggested different tools that can be used for developing better product. For example Poppendieck and Propendieck (2003) describe briefly about perceived integrity and conceptual integrity, and how relevant stakeholders are involved in each areas. In ASD, technical excellence only concern on developers and testers. But LSD concern widely for technical excellence where concerning on developers and testers are a part of it. It also talks about how to come up with a best solution using set-base development tool.

Leadership is one of the key stakeholders generated from LSD that is not present in ASD. Yes we have identified scrum master from Scrum for ASD but the roles and responsibilities between these two stakeholders are totally different. It is necessary to visualize the current work where LSD has a tool JIT (Kanban board) that has been used for work visualization. But for ASD there is not such opportunity rather than need to wait until the next iteration meeting. To achieve these practices we identified subject matter experts, users, customers, product owner, developers, master developer, operators, maintainers, analyst and testers are the key stakeholders in LSD.

7.1.2 Stakeholder theory Vs Agile and Lean stakeholders

In this section we will compare how stakeholder theory works with agile and lean principles separately. From stakeholder theory we identified stakeholders could be primary stakeholder or secondary stakeholder. Freeman et all (2008) identified customers, employees, suppliers, communities and financiers as a primary stakeholder and government, competitors, media, environmentalist, corporate critics and special-interest groups as a secondary stakeholder. As any of the stakeholders may become primary or secondary stakeholder based on product so rather than mentioning them as primary or secondary stakeholder here we will choose one by one stakeholder and we will see how each of them involve with agile and lean principles.

Stakeholder theory Vs Agile stakeholders

- **Customers**

Customers are one of the key stakeholders from stakeholder theory where agile methods concern deeply on customer satisfaction. Almost all of the agile principles are involved with customers. In agile literature we identified customers are considered as a general word where stakeholder theory suggested customers may vary from customers to customers for example customers could be singles, families, corporate customers or repeated customers. We belief that considering these variation of customers will help the company/project managers to think differently based on different arrival customers.

- **Employees**

Another key stakeholder from stakeholder theory is employees where employees are those people who are being paid from the company. Stakeholder theory suggested that employees could be executives, administrative staffs, middle managers, support staffs and entry level employees. While comparing with agile principle we identified agile methods do not involve all the group of employees with its principles rather than agile methods consider executives (includes senior manager, middle-management, project manager) and teams (includes programmers, requirement analysts, designers, testers, scrum masters, product owners, technical lead/chief architect, technical writes and so on) as a group of employees. One of the key agile principles is "Stakeholder involvement" where other stakeholders like human resource, finance, contracts etc are involved but the roles and responsibilities for those stakeholders are not clear. Moreover other employees groups like sales team, marketing team, operation team etc are also not involved in agile principles.

- **Financiers**

According to stakeholder theory financiers are those who are responsible for financial support. In stakeholder theory it is also known as owner. Stakeholder theory suggested that financiers could be class A stakeholder, class B stakeholders, bondholders or bank where in ASD financiers concept is collectively known as product owner. ASD encourage product owner's involvement in different principles where customers or project managers sometimes perform (based on size and importance of product) as a representative of product owner.

- **Others**

The other stakeholders like communities, competitors, media, special-interest group, government, and consumer advocates groups are not directly involved with agile principles. But we cannot avoid these stakeholders because new stakeholders may generate in any time based on product which might vary from product to product. From stakeholder theory one of the key stakeholders is suppliers where in software development suppliers are considered as a part of team (Freeman et al, 2008). In ASD one of the key stakeholders are users where users are suggested to involve almost all of the principles, although users are not well discussed in stakeholder theory literature.

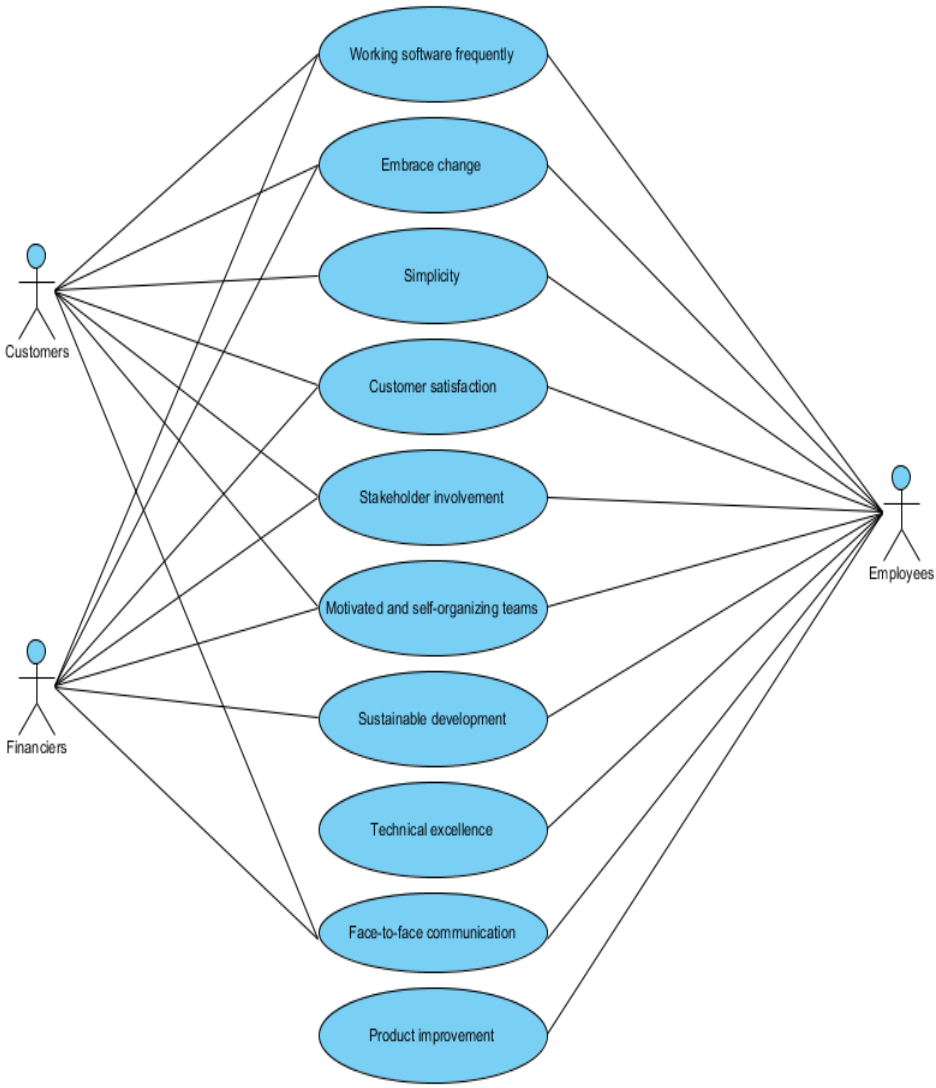

Figure 10: Stakeholder theory relationship with derived Agile principles

Stakeholder theory Vs Lean stakeholders

- **Customers**

Unlike agile principles several lean principles are directly focus on customers. Customers are also considered as a general word where variations of customers are not followed that is suggested by stakeholder theory literatures.

- **Employees**

While considering employees from lean literature it is not far away from agile principles. Unlike agile principle literature lean principles have executives (includes senior manager, middle-management, and project manager) and teams (includes developers, testers, requirement analyst, designers, document writers, domain expert, master developer/leader, operators, maintainers and so on); although the roles and responsibilities are not clearly mentioned for operators and maintainers.

- **Financiers**

While considering financiers in LSD it is not well focused comparing with ASD. Reason could be ASD focus on several team meeting (presence of product owner) where it is absence in LSD. Poppendieck and Cusumano (2012) suggested involving all stakeholders including financiers but the roles of financiers are not described well.

- **Others**

We identified communities, competitors, media, special-interest group, government, and consumer advocates groups are not involved directly with LSD principles.

Figure 11: Stakeholder theory relationship with derived Lean principles

7.2 Differentiating stakeholders based on Mitchell's model

The degree of importance or urgency and who and what really counts to the organization varies. As Mitchell, et al. (1997) stated we can measure the impact or perception of the stakeholders by categorizing them into three parts, the stakeholders' power, legitimacy and urgency. As it shown in fig 5, the three circles overlap and further classified into seven levels of perceived values of the stakeholders. As far as ASD and LSD concerned both have different stakeholders with varying degree of salience. In order to evaluate and compare the importance or impact of the different stakeholders in ASD and LSD, we will use the above paired principles that we used to explore the difference. *People focus v stakeholder involvement; motivated and self-organizing teams; sustainable development*: *people focus* in LSD involves stakeholders such as; project manager, designers, programmers, testers, domain expert, management, customers. Under this LSD principle we consider management and customers to be definitive stakeholders; project manager, programmers, testers and designers to be expectant stakeholders, while domain expert can be considered as latent stakeholder. Nearly similar situation occurs in ASD when we consider *stakeholder involvement, motivated and self-organizing teams;* the stakeholders can be programmers, system architects, technical lead/ chief architect, technical writers, management, customers, end-users, product owner, requirement analyst, human resource, finance, contracts, scrum masters. Among these stakeholders, definitive stakeholders can be product owners, customers, programmers, management, and scrum masters; where as system architects, technical lead/ chief architect, technical writers, requirement analyst can be considered as expectant stakeholders; and latent stakeholders could be human resource, finance, contracts and end-users. Moreover, under the ASD principle sustainable development, we categorize management and sponsors under definitive stakeholders, where expectant stakeholders can be programmers, testers, requirement analyst, and designers while users under latent stakeholders.

Customer focus V *customer satisfaction; face-to-face communication*: LSD's principle *customer focus* involves definitive stakeholders like customers and management. Where as, developers, designers, testers and requirement analysts are included under expectant stakeholders. When we come to ASD's two principles, *customer satisfaction and face-to-face communication,* we categorize customers, management, project manager, and product owner under definitive stakeholders; expectant stakeholders could be programmers, testers, requirement analyst, designers and scrum masters with end user is put under latent stakeholder.

Creating learning environment is one of the mottos of LSD and it involves its own expectant stakeholders like programmers, requirement analysts, testers and designers with partners to be latent stakeholders. Concerning *product improvement* in ASD, we have project manager, developers and scrum masters as definitive stakeholders while we consider requirement analyst, designers and testers to be expectant stakeholders.

Working software frequently Vs *Move towards flow and deliver values fast:* ASD is well known by its principle of delivering *working software frequently*, under this principle we categorized Programmers, customers, product owner, management and scrum masters as definitive stakeholders, whereas expectant stakeholder could be requirement analyst, designers, testers and last but not least end users are consider latent stakeholders. On the other hand LSD'S principles (*move towards flow and deliver values fast)* could include programmers, project manager, requirement analyst, testers, designers as definitive stakeholders, whereas expectant stakeholders could be customers with finance/accountant can be latent stakeholder since they have little to contribute.

Embrace change Vs Continuous improvement & decide as late as possible: *embrace change* in ASD involves definitive stakeholders such as customers, programmers, management, product owner and scrum masters, where as testers, requirement analyst and designers could be expectant stakeholders while end users can be latent stakeholders. When we go to ASD's principles (*Continuous improvement & decide as late as possible*), definitive stakeholders could be programmers, requirement analyst, designers and testers, where customers, managers and project manager could be expectant stakeholders.

Simplicity Vs Define customer value by eliminating waste: *Simplicity* in ASD involves programmers as definitive stakeholders and customers as expectant stakeholders. Whereas in LSD, *Define customer value by eliminating waste* considered as the main principle and could involve customers as definitive stakeholders and developers, testers, requirement analyst, designers, document writers, management and project manager as expectant stakeholders.

Technical excellence V product excellence: This are two similar principles from both sides, with t*echnical excellence* from ASD involves definitive stakeholders such as Programmer, tester, scrum master and designers, while *product excellence* from LSD involves developers, master developer, analyst and testers as definitive stakeholders while expectant stakeholders could be customers, operators, maintainers and domain experts, with users being latent stakeholders.

Table 3: Identifying stakeholders based on Mitchell's model

Agile				Lean			
Principles	**Stakeholders**			**Principles**	**Stakeholders**		
	Definitive	**Expectant**	**Latent**		**Definitive**	**Expectant**	**Latent**
Working software frequently	Programmers, customers, product owner, management .and scrum masters	Requirement analyst, designers, testers	End users	Move towards flow and deliver values fast	Programmers, project manager, requirement analyst, testers, designers	Customers	Finance/accountant
Embrace change	Customers, programmers, management, product owner and scrum masters.	Testers, requirement analyst and designers.	End users	Decide as late as possible	Programmers, requirement analyst, designers and testers.	Customers, managers and project manager.	
				Continuous improvement			
Simplicity	Programmers	Customers		Define customer value by eliminating waste	Developers, testers, requirement analyst, designers, document writers, management.	Project manager	
Customer satisfaction	Customers, management, project manager, and product owner.	Programmers, testers, requirement analyst, designers	Scrum masters, end user	Customer focus		Developers, designers, testers and requirement analysts.	
Face-to-face communication							

Agile				Lean			
Principles	**Stakeholders**			**Principles**	**Stakeholders**		
	Definitive	**Expectant**	**Latent**		**Definitive**	**Expectant**	**Latent**
Stakeholder involvement	Product owners, customers, programmers, management, and scrum masters.	System architects, technical lead/ chief architect, technical writers, Programmers, testers, requirement analyst.	Human resource, finance, contracts and end-users, Designers, end users.	People focus	Management, customers	Project manager, programmers, testers and designers.	Domain expert
Motivated and self-organizing teams							
Sustainable development							
Face-to-face communication							
Technical excellence	Programmer, tester, scrum master and designers			Product excellence	Developers, master developer, analyst and testers	Customers, operators, maintainers and domain experts	Users
Product improvement	Project manager, developers and scrum masters	Requirement analyst, designers and testers		Create learning environment		Programmers, requirement analysts, testers and designers.	Partners

7.3 Summary

In this chapter we have compared Agile versus Lean principles according to Weber's notion of ideal types. Based on established well- cited literature we have reconstructed the ideas behind Agile and Lean. We believe that rather than considering Agile or Lean principles separately a manager can benefit while combining Agile and Lean principles. For example requirement changes are always welcome in Agile but Lean also describe how to minimize requirement changes, how to achieve well requirements at the beginning of requirement gathering which will also minimize requirement changing at the end. In this chapter we have compared several Agile and Lean principles, rather than considering them the difference between Agile and Lean managers might be beneficiated while considering the comparisons themselves as a set of principles.

We have also identified the relevant practices and stakeholders under each principle. Agile has several practices like scrum meeting, sprint, sprint review etc where most of the stakeholders should be presented. What will happen if end user doesn't attend in the meeting or there might be no existence of end user if the size of the project is too small? What will happen if development team need to prioritize the features and customer is not present on the meeting? To answer such questions we have used Mitchell's model where we have divided stakeholders under definitive, expectant and latent stakeholders. It is impossible to follow any practices with the absence of a single type definitive stakeholder. At the same time we are not ignoring latent stakeholders although their priority is less than definitive stakeholders. This is a general view of our analysis but in practice any latent stakeholder may become definitive stakeholder based on the importance of the project.

8. Findings and Conclusion

In the introduction we had the following two research questions:

1. What are the differences between agile and lean methods based on a stakeholder perspective?
2. How can managers benefit from combining agile and lean methods?

In this research we have distinguished between Agile and Lean methods. To make this research narrower we have considered only stakeholder perspective. Later on, we have again considered only principles basis from both side i.e., we have identified the stakeholders from Agile and Lean principles. To achieve the first research question's answer, first we have analyzed agile and lean principles based on different authors suggested principles although for agile methods we have directly used Agile Manifesto principles (see section 4.2 and 5.2 for details). Then we have derived our own Agile and Lean principles while considering the presence of stakeholders (see section 6.1 and 6.2 for details). Moreover, in section 6.3 has included two way comparison work. One comparison is between Agile versus Lean principles and another is stakeholder theory versus Agile/Lean stakeholders. We believe that managers might be benefited while considering these comparison works as a principle of software development. To achieve our analysis stronger we have analyzed stakeholder theory where by Freeman, Harrison and Wicks (2007) we have identified customers, employees, suppliers, communities and financiers as a primary stakeholder and government, competitors, media, environmentalist, corporate critics and special-interest groups as a secondary stakeholder. Although we did not find any direct relationship of communities, government, media, environmentalist, corporate critics and special-interest with our Agile/Lean stakeholder analysis (principle based). We are not ignoring these stakeholders because stakeholder's generation is a dynamic issue where stakeholder's creation depends on size, value and importance of the product.

While comparing Agile and Lean methods we have found that Agile methods take care more on people than process and tools, on the other hand Lean methods take care less on people than process and tools. Agile method's all principles evolve with only product development where Lean method's principles evolve with more than product development. In this research we did

not identify a large number of stakeholder differences between Agile and Lean methods while considering principles. One of the most important differences is creating leader derived from Lean principles. Agile methods believe that team should be self-motivated, self-organized where there is no room for leadership. But in reality how can we ensure that our team is always self-motivated and self-organized? To achieve this one solution is to build team with only experienced people. Another solution could be creating a leader in the team who will encourage team to become self-motivated and self-organized where involving experienced people is not mandatory (see 5.3.4). While considering the principles from both Agile/Lean side we have identified that few principles were nearly same but the way of thinking were different. We believe that rather than considering either Agile or Lean, considering both of them together might provide better performance in software development.

From stakeholder theory we have identified that customers are one of the key stakeholders where customers could be single, families, corporate customers or repeated customers. End users are one of the key stakeholders in software development where end users are considered as a part of customers. In stakeholder theory there are no clear explanations for end users but the importance of end user in software development is unavoidable. A huge numbers of users are now involved to perform various activities in software development for example modifying or creating software artifacts from simple customization to software functionalities. Ye and Fischer (2007) classified seven types of end users as pure end users, end user who customize, end user who write macros, end users who develop web applications, developers using domain-specific languages, data-intensive researchers, and software professionals. Costabile et al. (2008) also describe how to minimize the communication gap between software developer and end users.

We believe that principles are not only for a group of people who are directly involved for product development. In both Agile and Lean method we have identified that principles are mostly evolving with a specific group of stakeholders ie customers, product team etc although both Agile and Lean method mentioned involving all the stakeholders on their principles. But in reality it is really hard to indentify all the stakeholders as identifying stakeholders may vary from project to project based on size, importance and other issues. Besides it is also required to identify a proper way of communication with other stockholders (who may not involve directly with product development) while developing software.

In most of the organizations human resources (HR), marketing, sales support and maintenances are one of the major departments. Besides primary and secondary stakeholders it is also necessary to spread Agile/Lean knowledge

on these departments. If an organization develop product based on Agile/Lean method it makes sense to have Agile/Lean knowledge to its marketing, sales support and maintenances teams. Wijewardena (2011) describe tailoring conventional HR activities into Lean (Kanban) method for better performance.

Moreover, developing software now-a-days is more challenging than before. Most of the organizations are now involving with outsourcing products which minimize cost and time both. We also believe that it is necessary to concern on outsourcing as well while considering stakeholders; not only identifying as a stakeholder but also needed to identify a proper way of communication while developing software.

In software development more than 50% of IT budget spent on software maintenance but we did not identify any maintenance team involvement neither in Agile software development nor in Lean software development. So it is also required to concern on maintenance team while scattering Agile/Lean knowledge. Prochazka (2011) explains how maintenance teams can work in Agile and Lean way for better support.

8.1 Future work

Recently global software development (GSD) has been widely adapting by many software companies over the world where the most two reasons for adapting GSD are: increasing project size and pressure on maintenance cost with limited available employees (Nidhraa et al, 2012). Establishing GSD in the organization is not too easy, besides other risks one the most unavoidable risk is identifying stakeholders. Moreover, it is also required to identify a proper way of communication with the stakeholders. We are going to conclude our thesis by mentioning our future work:

1. Who are the stakeholders in Agile/Lean methods in GSD?
2. How to communicate with the identified stakeholders?
3. What are the roles and responsibilities for each identified stakeholders?

9. References

Åhlström P., 1998. Sequences in the Implementation of Lean Production. *European Management Journal*.

Amble,r S. W. and Lines, M., 2012. Disciplined Agile Delivery: A practitioner's guide to agile software delivery in the enterprise, http://www.agilemodeling.com/essays/communication.htm, accessed 7th November, 2012.

Ams technology blog. http://technology.amis.nl/2008/10/03/agile-software-development-the-principles-principle-8-agile-processes-promote-sustainable-development-the-sponsors-developers-and-users-should-be-able-to-maintain-a-constant-pace-indefinitely/ accessed 19th November, 2012.

Barraza, M. F. S., Smith, T & Dahlgaard-Park S. M., 2009. Lean-Kaizen public service: an empirical approach in Spanish local governments. *The TQM Journal*, Vol. 21 No. 2, pp. 143-167.

Beck, K., 2000. Extreme Programming Explained: Embrace change. Addison-Wesley.

Beck, K. 2005. Extreme Programming Explained: Embrace change. Pearson Education, Inc.

Beyer, H., 2010. User-Centered Agile Methods. Morgan & Claypool Publishers series.

Bruuna, P., Meffordb, R. N., 2003. Lean production and the Internet. *International Journal of Production Economics*.

Calderone, C., 2010. http://www.hmetalk.com/forum/blogs/chriscalderone/2-thinking-lean-power-standardized-work-practices.html accessed 20th november, 2012.

Carvalho, R., Alves, A., and Lopes, I., 2011. Principles and Practices of Lean Production applied in a Metal Structures Production System. *International journal of Agile management system*.

Chena, J.C., Li Y. & Shady, B.D., 2010. From value stream mapping toward a lean/sigma continuous improvement process: an industrial case study. International Journal of Production Research, Vol. 48, pp. 1069-1086.

Cockburn, A., 2002. Agile Software Development. Pearson Education, Inc.

Cohn, M., http://www.mountaingoatsoftware.com/scrum/daily-scrum, accessed 7th November, 2012.

Conboy, k., 200. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. Information Systems Research,Vol. 20, No. 3, pp. 329–354.

Cooke, J.L., 2010. Agile Principles Unleashed: Proven approaches for achieving real productivity gains in any organization.[e-book]IT Governance Publishing. Available through: Uppsala University Library website <http://site.ebrary.com.ezproxy.its.uu.se/lib/uppsala/docDetail.action?docID=10438091>[Accessed 9 October 2012].

Costabile, M.F., Mussio, P., Provenza, L. P., Piccinno, A., (2008), "End Users as Unwitting Software Developers", ACM journal.

Cronin, P. Ryan, F. Coughlan, M., 2008. Undertaking a literature review: a step-by-step approach. <http://www.cin.ufpe.br/~in1002/leituras/2008-undertaking-a-literature-review-a-step-by-step-approach.pdf>. [Accessed online, on February 2013].

Donaldson, T. & Preston, L.E., 1995. The Stakeholder Theory of the Corporation: Concepts, Evidence, and Implications. *Academy of Management Review*, Vol. 20, No. 1, pp. 65-91.

Ebert, C., Abrahamsson, P., Oza, N., 2012. Lean software development. *IEEE Computer Society*.

Emiliani ,M. C., 1998. Lean behaviors. Management Decision 36/9 [1998] 615-631. MCB University Press.

Fowler, M. and Highsmith, J., 2001. The Agile Manifesto. http://www.pmp-projects.org/Agile-Manifesto.pdf, accessed 2nd November, 2012.

Freeman, R. E., Harrison, J.S. and Wicks, A. C., 2008. Managing for stakeholders: survival, reputation and success. [e-book]Yale University Press. Available through: Uppsala University Library website <http://site.ebrary.com.ezproxy.its.uu.se/lib/uppsala/docDetail.action?docID=10315690>[Accessed 9 October 2012].

Freeman, R.E. & Reed, D.L., 1983. Stockholders and Stakeholders: A New Perspective on Corporate Governance. *California Management Review*, Vol. XXV, No. 3, p. 88-104.

Freeman, R.E., 1984. Strategic Management: A stakeholder Approach. Pitman Publishing Inc.

Freeman, R.E., Harrison, J.S., Wicks, A.C., Parmar, B.L. & Colle, S.D., 2010. Stakeholder theory: The state of the art. Cambridge university press.

Highsmith, J., 2002. Agile software development ecosystem. Addison Wesley.

Hoda, R., Noble, J., Marshall, S. 2011. Developing a grounded theory to explain the practices of self-organizing Agile teams.

Holden, R. J., 2010. Lean Thinking in Emergency Departments: A Critical Review. Annals of Emergency Medicine, Vol. 57 No. 3, pp. 266.

Ikuma, L. H., Nahmens, I., and James, J., 2011. Use of Safety and Lean Integrated Kaizen to Improve Performance in Modular Homebuilding. *Journal of construction engineering and management*, Vol. 137, pp 551.

Jalali, S. & Wohlin, C., 2010. Agile Practices in Global Software Engineering - A Systematic Map. 2010 *International Conference on Global Software Engineering*, 2010, pp. 45-54.

Järvinen, P. (2000) Research Questions Guiding Selection of an Appropriate Research Method, in Hansen, Bichler and Mahrer (eds.), *Proceedings of the 8th European Conference on Information Systems*, 3–5 July, 2000, Vienna, pp. 124–131

Karlsson, C., Åhlström, P., 1996. Assessing changes towards lean production. *International Journal of Operations & Production Management*.

Koch, A.S., 2005. Agile software development: Evaluating the methods of your organization. [e-book]Artech House. Available through: Uppsala University Library website <http://site.ebrary.com.ezproxy.its.uu.se/lib/uppsala/docDetail.action?docID=10082017>[Accessed 10 October 2012].

Kohrell, D., 2011. Agile Principle 6 - Face to Face Interaction, <http://tapuniversity.com/2011/02/09/agile-principle-6-face-to-face-interaction/> [Accessed 7 November, 2012].

Larman, C. and Vodde, B., 2009. Lean Primer.

Larman, C., 2004. Agile & iterative development: a manager's guide. Pearson education Inc.

Liker, J.K. and Morgan, J. M., 2006. The Toyota Way in Services: The Case of Lean Product Development.

Lim, S.L., Quercia, D. & Finkelstein, A., 2010. StakeNet: Using Social Networks to Analyze the Stakeholders of Large-Scale Software Projects. ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE), vol. 1, pp. 295-304.

Max Weber, stanford encyclopedia of philosophy, http://plato.stanford.edu/entries/weber/#IdeTyp: accessed online on June 10, 2013

McManus, J., 2004. A Stakeholder Perspective within Software Engineering Projects, *International Engineering Management Conference*, vol. 2, pp. 880-884.

Middleton, P. , 2001. Lean Software Development: Two Case Studies. Software Quality Journal, p 241–252.

Middleton, P. and Joyce, D., 2012. Lean Software Management: BBC Worldwide Case Study. IEEE.

Middleton, P., 2001. Lean Software Process.

Middleton, P., Flaxel, A. & Cookson, A., 2005. Lean Software Management Case Study: Timberline Inc. Lean Software Management Case Study: Timberline Inc. pp. 1-9.

Miller, L. M.., 2005. Lean Teams Developing the Team-Based Organization: The Skills and Practices of High Performance Business Teams. http://www.lmmiller.com/assets/docs/Lean-Teams.pdf accessed 12th November, 2012.

Miskelly, H., 2009. Improving Customer Satisfaction with Lean Six Sigma.

Mitchell, R.K, J.D. , Wood, B. R. & Agle, 1997. Towards a theory of stakeholders identification and salience: defining the principle of who and what really counts. *Academy of Management Review*, vol. 22, no4, p. 853-887.

Moayed, F. A. and Shell, R.L., 2009. Methodology And Theory: Comparison and evaluation of maintenance operations in lean versus non-lean production systems. *Journal of Quality in Maintenance Engineering*, Vol. 15 No. 3, 2009 pp. 285-296.

Naylor, J.B., Naim, M.M. and Berry, D., 1999. Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain. *International Journal of Production Economics*, vol 62, pp. 107-118.

Nidhraa, S., Yanamadalaa, M., Afzalb, W., Torkar, R., 2012. Knowledge transfer challenges and mitigation strategies in global software development-A systematic literature review and industrial validation. *International Journal of Information Management*, Vol. 33, pp. 333-355.

Ohno, T, 1988. The Toyota Production System: Beyond Large-Scale Production. Productivity Press.

Perera, G.I.U.S. & Fernando, M.S.D., 2007. Enhanced Agile Software Development - Hybrid Paradigm with LEAN Practice. *IEEE, Second International Conference on Industrial and Information Systems*.

Phillips, Robert A. 2011. Stakeholder Theory: Impacts and Prospects. Edward Elgar Publishing Limited, Cheltenham.

Polk, R., 2011. Agile & Kanban In Coordination. *IEEE Computer Society*, pp. 263-268.

Poppendieck, M. & Cusumano, M.A., 2012. Lean Software Development: A Tutorial. *IEEE Computer Society*.

Poppendieck, M., Poppendieck, T., 2003. Lean Software Development: An Agile Toolkit". Addison Wesley.

Prochazka, J., 2011, Agile Support and Maintenance of IT Services. Information Systems Development.

Putnik, G. D., Putnik, Z, 2012. Lean vs agile in the context of complexity management in organizations. The Learning Organization, Vol. 19 Iss: 3 pp. 248 - 266.

Rico, D.F., 2010. Lean and Agile Project Management: For Large Programs and Projects. Business & Information Systems Engineering Journal

Rico, D.F., Sayani, H.H., & Field, R.F., 2008. History of Computers, Electronic Commerce, and Agile Methods. Advances in Computers, p. 1-55.< http://davidfrico.com/rico06d.pdf>. [Accessed online, on February 2013].

Rooney, D., 2011. http://practicalagility.blogspot.se/2011/05/simplicity-planning-and-weather.html. [Accessed 2nd November, 2012].

Rudolf, H. and Paulisch, F., 2010. Experience Report: Product Creation through Lean Approaches.

Scherrer-Rathje, M., Boyle, T.A. & Deflorin, P., 2009. Lean, take two! Reflections from the second attempt at lean implementation. Business Horizons, vol. 52, pp. 79-88

Schwaber, K. & Beedle, M. 2002. Agile Software Development with Scrum. Prentice-Hall, Inc.

Senge, Peter M., 1990. The Fifth Discipline: The Art and Practice of the Learning Organization.

Turk, D., France, R., 2005. Assumptions Underlying Agile Software-Development Processes. *Journal of Database Management.*

Wang, X., 2011. The Combination of Agile and Lean in Software Development: An Experience Report Analysis. 2011 Agile Conference.

Wanga, X., Conboyb, K., Cawleyc, O., 2012. Leagile software development: An experience report analysis of the application of lean approaches in agile software development. The Journal of Systems and Software, vol. 85, pp. 1287-1299.

Wijewardena, T., 2011. Do you dare to ask your HR Manager to practice KANBAN: The experience report of an offshore software company in Sri Lanka introducing agile practices into its Human Resource (HR) department. IEEE.

Womack, J.P. & Jones, D.T., 2003. Lean thinking: Banish waste and create wealth in your corporation. Newyork, NY, free press.

Womack, J.P. & Jones, D.T., 2003. Lean thinking: Banish waste and create wealth in your corporation. Newyork, NY, free press.

World Bank, 1996. The World Bank Participation Sourcebook. Washington D.C.: The World Bank

Yavuz, M., 2010. Fuzziness in JIT and Lean Production Systems. *Business & Information Systems Engineering.*

Ye Y. and Fischer, G., 2007. Designing for Participation in Socio-technical Software Systems. C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2007, LNCS 4554, pp. 312-321.

Appendix: Glossary of Terms

Agile a project management tool, guided by a manifesto of values and sets of principles

Backlog See Product Backlog

Cost of delay is brief calculation of cost and profit before taking any decision

Cycle time is the average time it takes something to get from one end of a process to the other

Daily scrum meeting is a 15 minute ritual that the team performs in each working day, to talk about the status of the work they are doing

Flow means to make the flow of values and ignore making any delay while value-adding activities and develop product "once at-a-time"

Heijunka leveling workload, smooth production, reducing waste

Jidoka means quality at the source. Automation, machine should serve for the people only

Just-in-time (JIT) production and delivery of the item at the right time with the right quantity

Kaizen continuous improvement to establish smooth flow

Kanban is a signal card to indicate the time to implement activities in value stream

Kano analysis a model used to increase product development and customer satisfaction

Lean is a management philosophy that focuses on elimination of waste, improving qualities, reduces delivery time and increase product improvement

Muda is a term used to express waste

Product Backlog consists of all the functionalities to be built into a system and it's prioritized by the product owner

Pull system means only develop the required products required by customer

Push system means the products are already produced before they are ordered

Queuing Theory is a mathematical method which is used for calculating delay of waiting in a line

Retrospective is a meeting involving all project participants to discuss the success of the project and other important matters

Scrum an iterative and incremental development method that gives more emphasis to "project management values and practices, rather than those in requirement, implementation and so on"

Scrum Master is responsible for reinforcing the project vision and goals; making sure scrum values, practices and rules are followed by each project stakeholder mediates between scrum team and management; removes any obstacles and tracks progress; conducts the daily scrum and the sprint review

Scrum Roles the stakeholders play one of the three roles; product owner, scrum master, and scrum team

Scrum team development team and they are responsible for working and achieving the sprint goal

Sprint/Iteration is fixed period of time, usually 30 days, that the team works on certain feature of the project in order to come up with an executable product increment

Sprint Planning a meeting between the team and the product owner to decide what functionality to build and how they do it

Sprint Review a meeting that the team presents the product increment that it has builds during the sprint

Stakeholder anyone who can affect and be affected by the achievement of organization's objective

Story detailed feature descriptions of product requirements, written on index cards and put on wall

Takt time/Common tempo means speeding work rate according to customer demand

Value stream the sequence of processes from supplier to customer while developing a product

Value stream mapping analysis and design the workflow in order to deliver the product to the customer

Waste is anything that doesn't add value to a product and that doesn't satisfy the customer's need

XP (extreme programming) focuses on the whole team working on a common reachable goal using its values and principles