Application of LEAN and BPR principles for Software Process Improvement (SPI):
A case study of a large software development organization

Master (MBA) Thesis
Revised Version

At Blekinge Institute of Technology, Sweden
September 2014

Thesis submitted to the
The Department of Industrial Economics and Management, Blekinge Institute of Technology (BTH)
in partial fulfillment of the requirements for the MBA degree.

Submitted by:
Umair Azeem Ansari
Syed Umair Ali

Master Thesis Coordinator:
Dr. Urban Ljungquist
Abstract

**Background**
Like other businesses, the failures and problems faced by the software development industry over the time have motivated experts to look for software process improvement to create quality software rapidly, repeatedly, and reliably.

**Objective**
The purpose of this study is to evaluate if and how Lean thinking and principles primarily associated with auto manufacturing industry can be applied to software development lifecycle for Software Process Improvement (SPI). The secondary aim is to analyze how BPR can be integrated with Lean software development for process improvement.

**Method**
A derived Lean-BPR adoption pattern model is used as a theoretical framework for this thesis. The seven Lean software development principles along with four-step BPR process are selected as process improvement patterns, which effects the KPIs of a software organization. This research study incorporates both Qualitative and Quantitative methods and data to analyze the objectives of this study. The methodological framework of Plan-Do-Check-Act is used in the case study to implement process re-engineering incorporating Lean and BPR principles. The impact of adopting the Lean and BPR principles is assessed in terms of cost, productivity, quality of products and resource management.

**Results**
Application of Lean and BPR principles for software process improvement in the organization under study resulted in **79%** improvement in test coverage, **60%** reduction in time for test execution and analysis and **44%** reduction in cost for fixing defects that were being passed to customer in past.

**Conclusion**
Based on case study results, it can be concluded that Lean, a bottom up approach, characterized by empowerment of employees to analyze and improve their own working process can be effectively combined with IT centric traditionally top down BPR approach for improving KPI’s and software processes.

**Key words:**
Acknowledgments

This Master's thesis is dedicated to our lovely families who have been very patient and supportive while we went through MBA program.

We very much like to thank our Master Thesis coordinator Dr. Urban Ljungquist for his valuable examination review comments on our first submission of thesis earlier this year in Feb 2014. This revised version of thesis is benefited tremendously in terms of thesis structure and content from his experienced feedback. Also we thank our previous MBA program coordinator Marie Aurell for providing us the clear and concise guidelines for writing this thesis.

Our special thanks to a person who made it possible for us. Konstantin Dolgov our internal supervisor at Motorola Solutions A/S Denmark for giving us the opportunity to work on our thesis in a highly dynamic environment and for all the logistic support. His technical insight and aspiring management made it easier for us to implement this project.

We owe to thank Lean and Six Sigma specialists at Motorola Solutions, James Joseph Waskiel and Issoufou Ba Hamidou for motivating us to take this initiative of restructuring the test processes in a large and complex organization.

We would like to thank all our teachers at BTH as well as classmates who made up for their physical absence through active and constructive discussions on online forums.

Umair Azeem Ansari
Syed Umair Ali
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPR</td>
<td>Business Process Re-engineering</td>
</tr>
<tr>
<td>CAT</td>
<td>Customer Acceptance Team</td>
</tr>
<tr>
<td>CMM</td>
<td>Capability Maturity Model</td>
</tr>
<tr>
<td>CMMI</td>
<td>Capability Maturity Model Integration</td>
</tr>
<tr>
<td>FL</td>
<td>Feature Load</td>
</tr>
<tr>
<td>FRB</td>
<td>Failure Review Board</td>
</tr>
<tr>
<td>FTR</td>
<td>Formal Technical Review</td>
</tr>
<tr>
<td>IPTs</td>
<td>Integrated Product Teams</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>IXIA</td>
<td>Hardware Network Traffic Generator tool</td>
</tr>
<tr>
<td>JIT</td>
<td>Just in Time (development)</td>
</tr>
<tr>
<td>KPIs</td>
<td>Key Performance Indicators</td>
</tr>
<tr>
<td>ML</td>
<td>Maintenance Load</td>
</tr>
<tr>
<td>NVA</td>
<td>Non Value Add</td>
</tr>
<tr>
<td>PDCA</td>
<td>Plan-Do-Check-Act</td>
</tr>
<tr>
<td>SCM</td>
<td>Supply Chain Management</td>
</tr>
<tr>
<td>SIT</td>
<td>System Integration Team</td>
</tr>
<tr>
<td>SPI</td>
<td>Software Process Improvement</td>
</tr>
<tr>
<td>TDD</td>
<td>Test Driven Development</td>
</tr>
<tr>
<td>TETRA</td>
<td>Terrestrial Trunked Radio</td>
</tr>
<tr>
<td>TPS</td>
<td>Toyota Production System</td>
</tr>
<tr>
<td>VA</td>
<td>Value Add</td>
</tr>
<tr>
<td>VSM</td>
<td>Value-Stream Mapping</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Introduction ................................................................................................................ .......................... 8  
  1.1 Background ................................................................................................................ ................... 8  
  1.2 Discussion of the Problem ................................................................................................. ........... 9  
  1.3 Problem formulation and purpose ........................................................................................... .. 11  
  1.4 Limitations and Delimitations ............................................................................................. ........ 11  
  1.5 Thesis structure .......................................................................................................... ................. 11  
2 Theory ...................................................................................................................... ........................... 14  
  2.1 Lean Background ........................................................................................................... .............. 14  
  2.2 Lean Software Development ................................................................................................. ..... 17  
    2.2.1 Agile and its association with Lean Software Development ............................................... 18  
    2.2.2 Lean Software Development Principles .............................................................................. 20  
  2.3 Business Process Re-engineering (BPR) ..................................................................................... . 26  
  2.4 Theoretical Framework ..................................................................................................... .......... 28  
    2.4.1 Derivation of Lean-BPR Adoption Pattern Model ............................................................... 31  
3 Method and Material ......................................................................................................... ................. 36  
  3.1 Stages for thesis project ................................................................................................. ............. 36  
  3.2 Research methodology selection ............................................................................................ .... 36  
  3.3 Data collection strategy .............................................................................................................. 37  
    3.3.1 Archival data collection strategy ....................................................................................... .. 38  
    3.3.2 Interview strategy ...................................................................................................... ......... 38  
  3.4 Reliability ............................................................................................................... ...................... 38  
    3.4.1 Form of bias ............................................................................................................ ............ 39  
    3.4.2 Validity and generalization ................................................................................................. 39  
4 Case Study .................................................................................................................. ......................... 41  
  4.1 Organization’s Background ................................................................................................. 41  
  4.2 Organization’s Product Development Model & Apparent Challenges ....................................... 41  
  4.3 Results of Survey and Informal Interviews session ............................................................... 43  
  4.4 Implementation details of Lean-BPR adoption model ............................................................. 46  
  4.5 Empirical results obtained after implementing Lean-BPR adoption model ......................... 54
List of Figures

Figure 1: Thesis overall structure ............................................................................................................ 12
Figure 2: Lean Principles (Source: Womack and Jones, 2003) .............................................................. 15
Figure 3: Comparison of Waterfall development model with Lean Software development along with Agile practices ......................................................................................................................................... 20
Figure 4: Mapping of seven wastes of manufacturing to software development (Mary & Tom Poppendeick, 2003) .................................................................................................................... 22
Figure 5: Coplien's organizational patterns for Agile software development (Coplien and Harrison, 2004) ....................................................................................................................................... 28
Figure 6: Relationships among the 5 patterns (in blue boxes) and 4 anti-patterns (in orange boxes). Van Hilst and Fernandez's (2010) system of Patterns and Anti-patterns ................................................. 29
Figure 7: Mapping of five patterns to existing process improvement frameworks (Van Hilst and Fernandez, 2010). ........................................................................................................................................... 29
Figure 8: Relationship among six patterns of Van Hilst and Fernandez's (2011) system of Patterns ........ 31
Figure 9: The Devil's Quadrangle (Dumas et al., 2013) .......................................................................... 32
Figure 10: Key Performance Indicators for Lean-BPR adoption pattern model ........................................ 32
Figure 11: Adopted patterns from Van Hilst and Fernandez's (2010, 2011) system of patterns ............... 33
Figure 13: Derived Lean-BPR adoption pattern model ............................................................................. 35
Figure 14: Motorola Solutions' Product Life Cycle as M-Gates .............................................................. 42
Figure 15: Cost in terms of $ and man weeks for a single defect found during Product Life Cycle (M-Gates) .................................................................................................................................................. 43
Figure 16: Legacy SIT test process work flow (As-Is state) .................................................................... 47
Figure 17: Current value stream map of SIT test process at FL1 time frame ........................................... 49
Figure 18: Future state value stream map of proposed SIT test process at FL1 time frame ................. 49
Figure 19: Re-Structured SIT test process work flow ............................................................................ 53
Figure 20: Conducted Test Cases per releases ......................................................................................... 54
Figure 21: Defect Detection in each releases .......................................................................................... 54
Figure 22: Time for Test Case Execution .................................................................................................. 55
Figure 23: Time spent on Test cases Execution vs. Tests result Analysis .............................................. 55
Figure 24: Defects raised (by downstream) per load per release ............................................................ 55
Figure 25: Estimated cost in terms of $ and man weeks ........................................................................ 56
Figure 26: Association of 22 Tools with 7 Leans Software Development principles (Source: Mary and Tom Poppendeick, 2003) ..................................................................................................... 78
Figure 27: Ford Example of Legacy accounts payable process (Source: Hammer, 1990) ................. 79
Figure 28: Ford example of re-engineered accounts payable process (Source: Hammer, 1990) ......... 79
Figure 29: Criteria for defining "Quick Win" at Motorola Solutions (Source: Lean Six Sigma Green Belt Training, 2008) ........................................................................................................................................... 80
List of Tables

Table 1: Comparison of Toyota Lean production with Microsoft Agile development in 1990s (Cusumano, 2010) ........................................................................................................................................................... 18
Table 2: Derivation of Lean-BPR adoption pattern model ............................................................................................................. 33
Table 3: Summarized findings of the case study project mapped with the patterns of Lean-BPR adoption pattern model ............................................................................................................................................. 57
1 Introduction
Successful businesses in modern world are characterized by optimal utilization of available resources leading to high speed product development and followed by error free delivery and customer satisfaction. Achieving these business goals for an organization becomes difficult in today's dynamic and aggressive market, if it adheres to its legacy business processes and organizational structure and overlooks the threats of competitive forces (Porter, 1990, 2008). It is also worth mentioning that some of the legacy business processes prevalent in organizations served the purpose well at the time they were initiated, but the changing needs, increasing competition and benefits offered by modern technologies render them inefficient and ineffective. This in turn limits the chances of increasing productivity and profitability of organizations.

In today’s highly competitive environment, continuous improvement is critical to any manufacturing company for establishing and developing quality processes and products. The traditional process improvement frameworks like TQM or Six Sigma etc. served the purpose well in past but recent times possess new challenges when organizations are limited in resources and need to be more lean and flexible (Glenn, 2009). The current situation demands for process assessment and improvement framework to be more light-weight, flexible and customizable to suit the needs of particular organization.

1.1 Background
A process can be defined as a sequence of activities and associated tasks required to meet organizational goals or create value for customers. An efficient process; irrespective of its area of implication e.g. administrative, financial, technical, or any other, orchestrate the flow of work across all the possible paths in order to meet the organizational goals. The tasks for the underlying process can then be performed manually or they require some computer interactions or can be completely automated (ITIS, 2010). In relation to that, process improvement activities are initiatives taken by organizations to make their processes more effective and efficient by adopting systematic approach to closing the process performance gaps e.g. by streamlining them, reducing the cycle time, identifying and eliminating the causes for nonconformance to customer’s requirements and specification etc. (ITIS, 2010)

There are many industry standard systems for process improvement such as Japanese Total Quality Control (JTQC), Total Quality Management (TQM), Business Process Re-engineering (BPR), Capability Maturity Model (CMM), Lean and Six Sigma to name a few. Since these systems originated in different industries as well as from different cultures, there are some inherent differences, which are apparent when it comes to their implementation. Yet they share the common goal of achieving improvement in quality of processes and products (Carey and Chiarini, 2011).

Pojasek (2005) classifies the most common process improvement models in to two broad categories, i.e. the traditional approach and the Systems Approach. In simplest of definitions, the traditional approach to process improvement stresses on “top-down” methodology where the
improvement program and its strategy has been planned and approved by the top management and the employees are involved only at a later stage where they are being told what they have to do. TQM, CMM, Six Sigma and BPR are example of such approach. On the other hand, Systems Approach emphasizes on carrying on process improvement initiatives from the “bottom up”. It motivates and encourages the actual personnel involved in the operational process to identify and seek improvements in their daily work routine with the help of variety of quality management tools. In most of the cases, the employees are trained to work efficiently with such tools for process improvement initiatives even though they might already be using them in other contexts. Lean methodology for process improvement is the prime example of Systems Approach. The Lean approach primarily used in manufacturing focuses on “aligning people, material, and information to deliver products with the shortest cycle-time, highest quality, and lowest cost” (Lean Six Sigma Green Belt Training, 2008). Lean is externally focused on meeting customer expectations and Value-Add (VA), while internally focused on encouraging its employees to eliminate sources of wastes in order to reduce costs in organization’s processes.

Like any other manufactured product, the quality of a software product is dependent on the underlying software development processes used for its creation (Viju, 2013). Software development process as a whole is considered to be highly complex and dynamic in nature due to fast changing customer’s needs and competitors threats. There exists numerous software development methodologies ranging from the traditional heavyweight and bureaucratic processes to lightweight agile processes, of which later have gained a lot of popularity in recent times (Abrahamsson et al., 2007). These traditional vs lightweight software development models forms the basis for either using “top-down” or “bottom up” process improvement initiative within an organization.

1.2 Discussion of the Problem

When it comes to quality of products and customer satisfaction, software development industry has been facing problems over the time. As per researches conducted in late 1990s and early 2000s, over 70% of software projects initiated worldwide were failing (Card, 2004). It was also reported that about 32% of all the corporate software development projects were cancelled before their completion date. In addition, almost 53% of all the software projects overrun their estimated cost by 189% (Linberg, 1999). A software project is considered a failure if it does not meet its budget, delivery and business objectives i.e. the customer value. All these failures and problems in turn prompted the software development industry to look for the ways to assess and improve its processes to create quality software rapidly, repeatedly, and reliably (Card, 2004).

For Software Process Improvement (SPI), organizations have been using several software specific process improvement methodologies such as CMM, CMMI, ISO 9000 as well as most recently started to benefit from the general process improvement standards mentioned above e.g. TQM, BPR and Lean etc. At times concerns are raised over software specific process improvement methodologies for focusing more on the improvement of management related activities rather than giving importance to process related activities and customer values. For instance, companies using CMM for their SPI are said to have so much focus on moving up to the next CMM level that they tend to forget the real goal of improving process with regards to customer value (Viju, 2013). Also conventional SPI methodologies e.g. CMMI and ISO etc. are proved to be beneficial for organizations only if they are initiated early in the software
development process; they are not designed to improve the process already in crises. They do not provide emergency solutions for recovering troubled processes. The legacy SPI methodologies not only require extra overhead in terms of excessive documentation but also require specialized resources in order to initiate, lead and manage these process improvement activities (Viju, 2013).

In recent years, the software development industry is fast moving towards adopting more lightweight development model such as Agile etc., which have been hailed as the silver bullet to address the issue of high software project failure rate. Many organizations who have been using the traditional development methodologies have either already adapted the Agile approach or in a transition to move towards it in order to seek its benefits which include but not limited to, fast time to market, quicker return on investment, better quality software products and higher customer satisfaction (Sidky et al., 2007). This transition also forces the software industry to focus more on the Systems Approach methodologies such as Lean etc. for improvement of software processes.

In spite of Lean’s potential for process improvement in software development, it is often discussed that “Development is not at all like manufacturing” and thus adopting Lean practices in software development is inappropriate. Poppendieck and Cusumano (2012) answered this dilemma with their novel research that if Lean is considered of as a set of principles rather than practices, it can be applied to software development lifecycle and can lead to process and quality improvements. Similarly, the concept of Lean Enterprise System is evolved over the years where different SPI methodologies can be brought together by exploiting their complementary relationships under Lean as core organizing framework (Bozdogan, 2010). Researchers promote such integrated enterprise system as not to find a unified solution that fit all needs but rather advocating a portfolio of mixed approaches for process improvement where the practitioners can pick the most appropriate approach according to their need and/or extends it to meet the improvement goals.

In recent years, different SPI approaches merged with Lean as underlying framework at an enterprise level are surfaced such as Lean Six Sigma (LSS) etc. Similarly, since its introduction in early 1990s, concept of BPR has undergone changes as well as it is experimented with supplementary process improvement tools and methodologies. Primarily, BPR focuses on integrating both business process restructuring and deployment of IT solutions to support the reengineering work. One of the emerging combination is to integrate BPR with customer centric Lean approach (Business Process in Six Sigma world, 2010). Incorporating Lean thinking, while a company undergoes radical changes in its processes means that new and re-engineered processes will lead to marked cost reduction by eliminating wastes and Non-Value Added (NVA) tasks and improves overall business process flow (Carey and Chiarini, 2011).

In general, most of the literature on the said subject of interconnected SPI approaches aimed largely at the practitioner audience and hence emphasize more on practical aspects rather than on building up theory-grounded methods (Bozdogan, 2010). While research has been done on specific implementations of Lean enterprise system, what is lacking is a holistic model of how Lean can act a central framework while integrating with other complimentary process improvement approaches e.g. BPR.
1.3 Problem formulation and purpose

The purpose of this study is to evaluate if and how Lean thinking and principles primarily associated with auto manufacturing industry can be applied to software development lifecycle for Software Process Improvement (SPI). The secondary aim is to analyze how BPR can be integrated with Lean software development for process improvement.

The problem of this thesis is formulated as:

*How can Lean principles be applied to software development and how can BPR, which is traditionally a top down approach, be implemented under a bottom up Lean framework for SPI?*

1.4 Limitations and Delimitations

This thesis is focused on the challenges and best practices related to adopting Lean principles in software development and hence along with theoretical discussions; implementation specifics such as relevant tools are discussed in relation to the subject matter. However, it is not within scope to define an entirely new methodology but rather to come up with a theoretical framework and steps that can help organization to transit towards Lean enterprise system incorporating BPR according to their particular needs.

While the results of this thesis are backed by a literature review, along with results from the case study incorporating surveys/interviews and implementation details, the actual benefit of the suggested framework cannot be quantified at this point of time. This would require several extensive case studies of other organizations applying our suggested framework into their software process improvement approach. This itself could be an area of further research in the emerging Lean SPI area.

Furthermore, practically, in the thesis case study, Lean and BPR initiatives are applied mainly on System Integration Test (SIT) team of a large software development organization. However, it is expected that other teams involved in overall software development lifecycle can adopt similar approach to overcome waste and NVA aspects to better utilize their resources.

1.5 Thesis structure

Diagrammatically, the thesis is structured in the following way:
Figure 1: Thesis overall structure
The thesis is classified into following chapters:

**Chapter 01:** This chapter provides the introduction and background of the said thesis project. It gives an insight of the problem and purpose of the thesis along with the structural overview of this thesis document.

**Chapter 02:** This chapter deals with the theories and reference studies as part of the literature review. At the end of this chapter, a proposed theoretical framework is derived from existing established models for process improvement along with the thesis hypotheses.

**Chapter 03:** This chapter lists the methods selected for conducting this research thesis along with the discussion of what other alternatives were considered. This chapter also discusses various reliability and validity concerns.

**Chapter 04:** This chapter gives an insight of the case study conducted based on the derived theoretical framework in chapter 02. It includes the data collected following the interview/survey process and case study implementation. Case study results’ general analysis concerning thesis hypothesis is also part of this section. At the end, all the issues and/or findings of the case study are summarized in tabular form where they are linked with the framework model attributes and relevant process improvement activities.

**Chapter 05:** This chapter details the analysis and evaluation performed on the case study results in relation to the research objectives of the thesis and in particular with the thesis theoretical framework. This chapter also generalize the findings of our case study with relevant research and literature.

**Chapter 06:** In the final chapter, we summarizes our conclusions and come up with relevant implications and recommendations.
2 Theory
This chapter summarizes the theoretical foundations for this thesis with a literature review of the core topics such as Lean in general, Lean in software development and BPR etc. As the thesis topic is quite practical in nature, there are also some practitioner journals, conference papers and white papers included in the literature review along with scholarly journals and academic sources.

A proposed theoretical framework follows the literature review for Lean and BPR incorporation for software process improvement, which is derived and/or adapted, from existing established models. This framework provides the basis for all the subsequent analysis and discussion in later chapters. Finally yet importantly thesis hypothesis based on the proposed theoretical framework are listed at the end of this chapter.

2.1 Lean Background
The concept of Lean originates from The Toyota Way or Toyota Production System, a philosophy developed and implemented by Toyota for Japanese automobile manufacturing since the late 1940s. This revolutionary approach helped them to produce automobiles with about half the number of labor hours as Ford and other leading US and European automakers of that time and most notably with higher quality and customer satisfaction (Poppendieck and Cusumano, 2012). Prior to that, Ford was ruling the automobile industry with cheap cars but at the expense of variety. Ford first developed the concept of mass production and the moving assembly line. Ford’s huge River Rouge and Highland Park plants in Michigan were example of continuous production flow with the help of raw materials produced in bulk and in-house. At the same time the installed single-purpose machines were able to manufacture the standardized products in very high volumes. In other words, as all the raw materials were flowed in one end and with the help of highly sophisticated machines; the Ford’s famous automobile Model T came out as an output at very fast rate (Krafcik, 1988).

This mass production was indeed the cheapest and fastest way to make cars, but it also meant manufacturing and storing a very large number of raw parts in hand, to make thousands of similar cars. With its benefits, there came also some repercussions such as the machinery used for production was expensive as well as very intolerant if mishandled. This in fact forced the mass producers to add many buffers in their production line in form of extra raw supplies and extra workers to name a few (Womack et al., 1990).

In late 1940s, a small Japanese automobile manufacturer Toyota first recognized the fact that for their small Japanese market the large number of cheap but similar mass produced cars is not needed. Taiichi Ohno, who also now known as the father of the Toyota Production System (TPS), experimented and mastered the whole new way to think about manufacturing and eventually product development. The central idea of his work is to eliminate waste from the processes and in doing so he actually gave an altogether new meaning to the word “waste”. Ohno was able to successfully change the mind-set inside Toyota by preaching the fact that “anything that does not create value for a customer is waste” (Marry and Tom Poppendieck, 2003). Toyota
employees accepted and understood this fundamental principle and started applying the concept by eliminating these wastes from their system. For them any unused item sitting around and waiting to be used was a waste. Similarly any part produced which was not needed immediately was also considered as waste. These acts actually enabled Toyota to reverse the flow of information signals and to control the production operations (Poppendieck and Cusumano, 2012). The paradigm shifted towards a “pull” system that actually used the exact amount of materials and components as needed for the final assembly and shipment through a continuous flow within the production system. In contrast to a “push” system, used by mass production manufacturers as Ford to stock the product based on predetermined production plans. Poppendieck and Cusumano (2012) also referred to the same concept as “Just in Time” (JIT) development, rather than “Just in Case” development of inventories.

Krafcik first used the word “Lean” opposite to the Ford-style “buffered” mass production system while carrying out his research work on global auto assembly plants at MIT in 1988. In the beginning of 21st century, Lean philosophy gained worldwide popularity and acceptance as a result of work conducted by MIT researchers (Womack and Jones, 2003).

Lean itself is not only a set of empirical tools or methodology but also most importantly a state of mind focusing on continuous improvement by continuous reduction or elimination of waste. Lean is classified as a bottom-up people-oriented approach based on empowerment of the employees who actually perform the tasks so that they can make critical and most value added improvements in their own domains. (Morgan, 2005). Improving flow and eliminating waste are two of the main drivers to become a Lean organization (Carey and Chiarini, 2011).

Lean is a holistic management approach about people, products and processes, adhering to following five fundamental principles (Malladi et al., 2011):

1. **Identify Value**: Understand and identify measurable values from the customer's standpoint. E.g., required features set and expected time frame of product delivery. In general, *Value* can be anything for which customers are likely to pay for.

2. **Map the Value Stream**: Identify each step of the process in detail. Mapping of process stream helps in identifying the inefficiencies within the process, areas of waste and non-
value-added activities. Value-stream mapping (VSM) is a pencil and paper exercise, which helps in understanding the flow of material and information as products move from their conception to deployment in their respective product development lifecycle. (Marry & Tom Poppendeick, 2003) Following order of steps are used for this exercise:

**Step 1**: Draw the current state map of activities showing the current steps, delays and information flows for certain process.

**Step 2**: Analyze and assess the current state map in terms of identified values from the customer and wastes in it.

**Step 3**: Draw a future desired state map with focus on value-added activities.

**Step 4**: Work towards achieving this future state by following the Lean principles (discussed next) and continuously assessing the current state with the desired future state.

3. **Create Flow**: Aligning the value-creating steps in the value stream in a way that allows the product to flow smoothly in overall product life cycle. It is a continuous effort to sequence the steps that provide the most efficient flow to meet the customer's demands.

4. **Establish Pull**: This implies that teams are able to begin and finish the tasks when it is needed by the next step downstream. In other words, let customers *pull* value or work from the upstream activity.

5. **Seek Perfection**: Establishing within organizational teams that it is everyone’s responsibility to identify value and understands current and future value streams, eliminate wastes in the steps, and introduce flow and pull. This continuous and relentless pursuit of perfection is the essence of Lean thinking in which ideally perfect value is created with no waste.

Lean is externally focused on meeting customer expectations and Value-Add (VA), as well as internally focused on eliminating the waste to reduce costs and increase quality in organization’s processes. Fujimoto (1999) illustrated following three source of wastes in manufacturing processes:

1. **Variability** (*Mura* in Japanese) can be of number of types such as varying cycle lengths, varying batch sizes of work packages, varying size of one work package, varying team members or size, varying delivery times, defects and irregular arrival of requests etc.

2. **Overburden** (*Muri* in Japanese) is often seen in the form of overtime by employees to meet arbitrary deadlines as well as over-dependence on experts for example only a single product manager knows several features in detail within his products portfolio

3. **NVA (Non-Value Add)** (*Muda* in Japanese) actions are of several types such as over production of solutions or features which are not required at that moment, duplication in work, waiting for clarifications or approvals, delayed delivery by other relevant teams etc. They are also commonly known as “seven forms of wastes” and are expressed as DOTWIMP acronym comprising of Defects, Overproduction, Transportation, Waiting, Inventory, Motion, Processing (Shingo, 1981).

Muda is classified further in to two types:
Type I Muda: are often called as necessary wastes, as they do not add value to customers directly, but are necessary as a business or system to function e.g. paperwork for compliance with government regulations etc.

Type II Muda: are the unnecessary wastes (DOTWIMP) and should be eliminated first.

Over the years, Lean thinking as set of principles have proven applicable to a wide range of industries. Although it was first started in manufacturing, but has evolved specially after 1980s to be applied to other diverse domains such as logistic, supply chain, healthcare and software development etc. to name a few. (Hibbs, 2009) (Waglay, 2013). Some of the examples showing benefits of adopting Lean thinking in various industry domains are listed in Appendix A. Jones and Mitchell (2006) summarized the typical benefits of adopting Lean principles regardless of their industry as following:

1. Improved quality and safety: prevent accidents and reduce errors resulting in quality product and services.
2. Improved delivery: reduce end-to-end delivery time without compromising on quality.
3. Improved throughput: the same resources (same people using the same equipment) are empowered and motivated to improve the way of their working to get improved results.
4. Accelerating momentum: an environment with standardized procedures promoting continuous improvement.

2.2 Lean Software Development

In recent years, the success of any business can be weighed in terms of profit they made as well as how much they have saved. This demand of cost saving as well as other uncertainties of business world has also motivated software industry to look for ways through which Lean thinking can be applied to the software development to increase its productivity and efficiency.

Researchers began to question the traditional software development approaches about their capability to address the changing needs of the software development industry in 1990s. For example, earlier it was assumed that complete set of software development requirements can be gathered at the beginning of the development lifecycle in order to eliminate change later. Whereas today it is not advisable to eliminate change early, rather it is recommended to handle inevitable changes throughout development lifecycle as an act of responsiveness to changing business needs and conditions. (Cockburn and Highsmith, 2001)

By mid 1990s, some similarities can be spotted between the then evolving software development approaches and the Japanese management style (Lean approach). Cusumano and Selby (1995) in their book Microsoft Secrets first noted the existent similarities between Microsoft’s daily build and Toyota’s JIT production approach. One of them being is that developers at Microsoft have to stop and fix known bugs on higher priority than to work on new feature development. This was named as “Sync and Stabilize” process (Cusumano and Selby, 1995). This approach has a striking resemblance with the Toyota’s way of working as workers are compelled to stop the assembly lines as soon as they detect problems and fix the issues immediately as first priority to ensure quality in the final product. Following Table 1 summarizes Cusumano’s findings regarding the similarities between the two said organizations in their way of working (Cusamano, 2010):
Table 1: Comparison of Toyota Lean production with Microsoft Agile development in 1990s (Cusumano, 2010)

<table>
<thead>
<tr>
<th>Toyota-style “lean” production (manual demand-pull with Kanban cards)</th>
<th>1990s Microsoft-style “agile” development (daily builds with evolving features)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JIT “small lot” production</strong></td>
<td>Development by small-scale features</td>
</tr>
<tr>
<td><strong>Minimal in-process inventories</strong></td>
<td>Short cycles and milestone intervals</td>
</tr>
<tr>
<td><strong>Geographic concentration—production</strong></td>
<td>Geographic concentration—development</td>
</tr>
<tr>
<td><strong>Production leveling</strong></td>
<td>Scheduling by features and milestones</td>
</tr>
<tr>
<td><strong>Rapid setup</strong></td>
<td>Automated build tools and quick tests</td>
</tr>
<tr>
<td><strong>Machine and line rationalization</strong></td>
<td>Focus on small, multifunctional teams</td>
</tr>
<tr>
<td><strong>Work standardization</strong></td>
<td>Design, coding, and testing standards</td>
</tr>
<tr>
<td><strong>Foolproof automation devices</strong></td>
<td>Builds and continuous integration testing</td>
</tr>
<tr>
<td><strong>Multiskilled workers</strong></td>
<td>Overlapping responsibilities</td>
</tr>
<tr>
<td><strong>Selective use of automation</strong></td>
<td>Computer-aided tools, but no code generators</td>
</tr>
<tr>
<td><strong>Continuous improvement</strong></td>
<td>Post mortems, process evolution</td>
</tr>
</tbody>
</table>

Earlier the term “Lean” was supposed to be just another name of Japanese management techniques but now it has evolved into a well-structured product development paradigm that emphasized on flexible and iterative development process with end-to-end focus on creating value for customer, reducing waste in terms of time and resources and empowering people for continuous improvement. The newly established set of techniques for software development under Agile umbrella possess these characteristics by promoting iterative, incremental and less sequential process than the traditional waterfall software model (Poppendieck and Cusumano, 2012). The waterfall model which was widely used for software development stresses on sequential phases starting from requirement gathering and then moving down to design, implementation, testing and deployment. This require each phase to be fully completed before moving on to the next phase. This approach focuses to eliminate the changes early in the software lifecycle. On the contrary, Agile advocates changes are deemed necessary during software lifecycle as they show the responsiveness of process for the changing business needs and put its emphasis on reducing the cost of change throughout a software development project. (Cockburn and Highsmith, 2001)

### 2.2.1 Agile and its association with Lean Software Development

By the end of 20th century, quite a handful of alternative software development methodologies surfaced claiming to be fulfilling the demands of modern and dynamic business needs. In February 2001, a group of like-minded software experts representing various evolving software development processes including Extreme Programming (XP), Scrum, Feature-Driven Development (FDD) and Dynamic Systems Development Method (DSDM) etc. had joined their hands together to produce the Manifesto for Agile Software Development. The manifesto outlines Agile’s way of working preferences over legacy software lifecycle practices. It states (Cockburn and Highsmith, 2001):

- *Individuals and interactions over processes and tools,*
• Working software over comprehensive documentation,
• Customer collaboration over contract negotiation,
• Responding to change over following a plan.

Traditional software development methodologies were generally based on all-inclusive rules i.e. all the things one should do under all situations. Whereas Agile as recommended in its manifesto, offers generative rules, i.e. a minimum set of things one must do under all situations to generate customized practices fulfilling special needs for an organization. This actually empower the team itself and promote creativity among team members to find smart ways to solve problems as they arise.

One of the major commonalities among various Agile processes is to have short iterative cycles and to combine them with the dynamic feature planning and prioritization in each of these cycles. Prioritization of various features during development enables customer to have the higher priority features developed and available up front. The novel research work conducted and published by Marry Poppendieck and Tom Poppendieck (2003) laid the foundation of what we now know as Lean Software Development. Their work and results not only popularized the Lean thinking in software development but also provide the basis to differentiate rather associate the term Lean with Agile for software product development.

One of the common question rather a misconception is “Aren't Lean and Agile just the same thing?” (Hibbs, 2009). The misconception is partly because Lean software development and Agile share most of their goals such as to increase the productivity of software development as well as the quality of the product itself along with the underlying processes applied. They both encourage empowering teams, adaptive planning as well as continuous improvement of the way teams are carrying out their tasks. It also adds the confusion when one can observe the practices under various Agile methodologies supporting the basic Lean principles.

However Agile has a different underlying perspective and philosophy from Lean and generally a limited focus when compared to Lean. Agile focuses mostly on the specific practices for developing software and the project management aspects of this development. It generally do not concern with the surrounding business context of development. On the other hand, Lean looks at the broader business context and provide a system of practices, principles and philosophy that creates value for end customers. (Poppendieck and Cusumano, 2012).

Figure 3 illustrates the comparison of the traditional Waterfall model with the Agile software development practices under direction of Lean software development principles:
Figure 3: Comparison of Waterfall development model with Lean Software development along with Agile practices

2.2.2 Lean Software Development Principles

Marry and Tom Poppendieck (2003) took the initiative to derive seven Lean software development principles from the established Lean fundamental principles in manufacturing. These can be directly applied to software development to achieve quality, savings, speed, and business value. They also presented 22 tools to actually implement these principles depending upon the unique situation of each software development organization (See Appendix B for their mapping). This toolkit facilitates the team to decide what development practices might be adopted, depending on the current situation and needs of the organization. The toolkit also includes various Agile processes and practices tied together with Lean software development
providing additional credibility to both approaches. These seven Lean software development principles, which are slightly modified over the years, are discussed in the subsequent sections (Poppendieck and Cusumano, 2012).

2.2.2.1 **Principle 01: Optimize the Whole**

An organization is comprised of various interdependent and interacting departments and teams. The ability of an organization to achieve its goals and purpose depends on how well all the underlying departments and teams interrelate and work together rather than measuring how well these teams are performing individually. This vary principle of lean software development is build on one of the first principle of Lean thinking i.e. to identify value. To understand and define measurable values with respect to customers is not a simple process when it comes to software development. This is primarily because software does not have any value of its own, but the value of software is judged in the context of a whole system e.g. as an automated car, or an auction website etc. The customer expectations are translated in to measurable metrics and are used to evaluate overall product performance.

The larger and complex the product and/or organizational structure is, there is more tendency to divide it into parts and then manage and control these parts locally. This eventually tends to create local measures of performance or local metrics. Any process improvement activity on these local metrics would result in local improvement but usually at the expense of the whole value stream, and therefore often resulting in decrease in overall performance. This pattern is referred to as suboptimization (Mary and Tom Poppendeick, 2003). It is therefore advisable to include as much of value stream as possible whenever there is a process improvement or optimization activity.

2.2.2.2 **Principle 02: Eliminate Waste**

As discussed earlier, eliminating waste is the fundamental principle of Lean thinking. The first step to implement Lean development is therefore learning to see the wastes. The second step is to uncover sources of waste and classify them depending on their impact and start eliminating them. The next step is to repeat the above mentioned steps, i.e. a continuous effort to eliminate the causes which do not create any value to the customer.

However, the definition of waste itself can be different depending on the unique situation. Shegeo Shingo (1981), who worked alongside Taiichi Ohno identified and listed seven types of manufacturing waste. The said list helped manufacturing managers around the world to find the hidden wastes and help eliminating them over number of years. Mary and Tom Poppendieck (2003) came to help the software development resources including managers and engineers by translating these seven established and defined manufacturing wastes in to software development perspective as shown in Figure 4.
1. **Defects:**
Defects require expensive rework to rectify the malfunctioning and is a non-value added waste. Lean software development emphasize on preventing defects as much as possible whereas traditional software development approaches inclined towards spending time and resources on defects detection. The statistics shows that in legacy software development processes, on average 40 to 50% of development time is consumed in finding and fixing defects (Poppendieck and Cusumano, 2012). The damage done by a single defect can be measured as a product of the impact caused by the defect and the time it remained in the product/software unnoticed and undetected. That means a critical defect which is detected much earlier is not a big threat or source of waste, than a minor defect which remains undiscovered in the system for long.

2. **Extra Features:**
The Pareto principal or the 80-20 rule in software development suggests 80% of customers’ needs are fulfilled by 20% of the product features (Hibbs et al., 2009). The remaining features are rarely or never used. This turns out to be major waste over time as it continue to eat up the project resources. During the software lifecycle, each line of written code has to be compiled, integrated, tested and maintained over and over-again and the extra code or features that cannot be tracked to real customer needs or requirement will add a potential failure point and a serious waste. If the code is not needed now, it is advisable not to include the “just in case” extra code.

3. **Task Switching:**
Involving same resources in multiple projects at the same time is a source of waste as it slows down the flow and productivity. Every time software developers switch between tasks or
projects, it takes considerable amount of time to get the mind focused on the current task by understanding the pre-requisite of it and then to get into the flow. While moving back and forth among tasks, software developers have to “relearn” every time to be at the same level where they left and be productive again (Hibbs et al., 2009).

4. **Delays:**  
Delays in software development are usually seen in situations where people are waiting for some actions to happen such as waiting for project initiation, waiting for documents to be created and reviewed, waiting for pending decisions, waiting due to test setups etc. From customer’s perspective, these delays in the development cycle increases the response time for their changing needs and hence prevent them from getting the value as quickly as they desired. This delay become critical if the software is being developed for an evolving domain e.g. for current smartphone market. The fundamental Lean software development principle of delaying a decision as late as possible to catch up with the uncertainty is to be assessed based on how quickly the development can be done once a decision is made.

5. **Partially done Work:**  
Partially done work is a work that has been started but was left somewhere in the middle i.e. not fully done. The example could be requirements pending for approval from customers for a certain feature, the undocumented design requirements, the requirements that are not implemented or coded yet, the untested code, the known defects that are not fixed and a completed code which is not deployed etc. There is a risk associated with the partially done work to become obsolete later in the development cycle if not needed likewise extra features waste discussed earlier. This can be viewed as investments already made in terms of resources, cost and time but waiting for the yield that may or may not occur.

6. **Handoffs:**  
Traditional software development approaches such as Waterfall requires moving various artifacts including requirement documents, design documents and test case documents etc. from one group to another sourcing a waste in form of handoffs. These documents cannot record everything that their specialist creators i.e. architects, programmers or testers etc. have learned and discovered while working. That resulted in a loss of large amount of tacit knowledge that remained with the creator and is not passed on or handed off to the next person or group in line. Hibbs et al. (2009) shows that the knowledge loss is compounded with each handoff i.e. suppose 50% of knowledge is lost in each handoff, testers will only receive 25% of the original knowledge handed off from designers and programmers. That is a loss of 75% of tacit knowledge in just two handoffs.

7. **Extra/unneeded Processes:**  
In some software development scenarios where the requirements are frequently changed, there is a need to synchronize the customer expectations with the code regularly. Mary and Tom Poppendieck (2003) suggests instead of having huge paperwork to deal with that situation, a value added activity could be to use table-driven or template driven condensed format for requirements that can be traced directly to written code as well as to the defined test cases. This will help not only the customers and developers to rapidly understand and trace the requirements
but also for testers to validate these changing requirements with their updated test cases. They also suggested defining and documenting condensed customer test cases instead of writing lengthy requirements. Another suggestion in this regard is to delay documenting the detailed feature requirements until the iteration where that feature will be implemented to reduce the rework effort.

2.2.2.3 Principle 03: Build Quality In
Taiichi Ohno’s idea of having quality within manufacturing processes is to make each step capable of finding and fixing its defects. In TPS, whenever the defect is detected, the entire auto manufacturing assembly line stops until the root cause of the problem is found and particular defect is fixed or prevention measures are taken care of. This quick fixing of defects stops introducing more related defects in the system and increase the overall quality of the product (Hibbs et al., 2009). Traditional software development approaches works similarly like pre-Lean era of automobile manufacturing industry, i.e. the defects remains in the system for long, introducing more defects, only to be detected in later stages of testing and quality control. However, at that stage, damage has already been done in terms of cost, resources and time required for rework.

The Lean approach is to make the code mistake-proof by writing and executing the tests in parallel while coding a particular feature. Test Driven Development (TDD) is a well known Agile concept, that encourages writing the feature related test cases prior to coding any feature and marked them all failed at the start. As you start coding the feature, these test cases can be passed one by one until all of them are passed to mark the completion of feature development.

In terms of processes, Schulmeyer (1990) introduces the idea of zero defects where the focus is on defect prevention carried out by extensive use of checklists, mistake proofing, metrics collection and inspections.

Lean approaches also discourages to wait till the last stage of development cycle to integrate small components of developed software in to a large system. Already discussed Microsoft’s synch and stabilize process, continuously integrate the small units of tested software into a larger system. This enables to detect the integration errors as soon as they are introduced rather than residing them in the system for long.

2.2.2.4 Principle 04: Learn Constantly
The relation between software development and manufacturing or production can be best explained with the example of creating a recipe and following the recipe. Both of these activities should be carried out with different approaches. Development as creating a recipe is a learning process involving several trials and errors. The very first attempt of a new recipe is not expected to be the perfect one. The development of a recipe requires trying many variations of ingredients and steps and then select the best method as a final one. On the other hand, manufacturing is like following the best recipe to repeatedly produce the same dish with a minimum of variation (Mary and Tom Poppendeick, 2003).

Lean software development envision development as continuous creation and integration of knowledge in to the product. Poppendieck and Cusumano (2012) recommends achieving this in
two different ways. First one as they named is “learn first” approach i.e. the development by exploring multiple options to solve or develop a particular feature from start. There can be many alternatives available to solve a given problem such as selection of programming language or 3rd party software components etc. The idea is to delay critical decisions late in the development cycle and then select the option that is most suited for the current situation based on the best available current knowledge. In this way, even at some later stages if the selected option proves not to be the optimized one, there is always the plan B or the next in priority option ready to be applied, as multiple options are already explored. This approach is also sometime referred as set-based design. The second approach is to start with a working prototype with minimal capabilities followed by multiple iterations and frequent delivery to the customers. Using customer feedback, build up the product by adding new as well as improving the existing features in the product. In this continuous leaning process, the final product will always have the most valued features. Both these two approaches can be combined together to achieve even better results by balancing the options and constraints for the critical decisions for the changing customer requirements over time (Poppendieck and Cusumano, 2012).

2.2.2.5 Principle 05: Deliver Fast

Lean principles emphasize not only deferring the decision until the last reasonable time to have more accurate knowledge and information but also stress strongly in delivering fast once the decision has been made. In a larger context, rapid delivery of developed software helps customers to increase their business flexibility. For some customers, fast delivery of developed software enable them to delay their own business decisions knowing they have quicker response and solution available. On the other hand, they can start yielding quickly on their investments.

In Lean software development’s context, delivering fast means developing prioritized features in short iterations and delivering them to customers for quick feedback. In legacy software development approaches such as Waterfall, one has to wait for the customer feedback until the very end of project when the customer actually start using the software. It is natural, until customers can see, touch or get the real feeling of any product (including software) by using it, they cannot fully realize their own needs. By following Lean software development approach, the features can be developed and delivered to customers before they start use their imagination to change their own needs and requirements. It helps customers to actually use these features and provide real feedback to change and reprioritize requirements. In this way, the final product meets the customer’s requirements more closely and eliminates the risk of large amount of rework (Peyton, 2009).

2.2.2.6 Principle 06: Engage Everyone

The most valuable resource for any organization is its human resource and their minds. Respecting them for their work, recognizing their individual and team accomplishments and actively seeking their advice are the motivation factors for them to be engaged with the vision of the organization (Hibbs et al., 2009). The trust showed by the employers or higher management towards employee that they themselves knows the best way to do their job, actually engaged them to expose discrepancies in the current processes and that in turn encouraged them to not only improve their own job functions but also the surrounding processes (Peyton, 2009).
Traditionally, a software development team is considered as a separate entity that is responsible for a software failure or success. Lean software development stresses the need of considering the software development as part of overall product flow and should be placed in line with other business units. Instead of isolating different teams, the value stream team is comprised of people from various background such as designers, developers, testers, operation and support teams, finance and marketing etc. whose collective understanding of customer and their needs are transformed in to a valuable final product (Poppendieck and Cusumano, 2012).

2.2.2.7 Principle 07: Keep Getting Better

Lean thinking encourages a constant quest on behalf of all the employees to improve the work system and processes with a firm belief that no matter how well the current systems is working, there exist an even better solution to the problem at hand (Poppendieck and Cusumano, 2012). Both the top leadership support and staff engagement are pivotal to the success of any continuous improvement activity (Anotny, 2011).

Today, many software organizations after adopting improvement programs such as CMM and ISO9000 etc. have faced difficulties in either implementing them or sustaining them. The main reason behind that result is actually the top-down nature of these improvement programs where the direction for improvement comes from the top management and employees are just religiously following the guidelines without using any of their own creativity. On the other hand, Lean creates an environment where the employees are encouraged to take the initiative and let the above management know the best way to accomplish tasks. The management responsibility in this case is to be supportive, take prompt action on their suggestions and analyze together the impact and value of the implemented changes (Mary and Tom Poppendeick, 2003).

Practically, Agile software development practices make continuous improvement mandatory as part of the routine work. For example, in Scrum, after end of each sprint (short 2-3 weeks iterations) review and retrospective sessions are mandatory rituals. The improvement ideas are discussed among the team members apart from the features or tasks to be worked on next iteration. The chosen improvement suggestions will also become part of backlog tasks for immediate or future iterations.

2.3 Business Process Re-engineering (BPR)

In 1990s, Business Process Re-engineering (BPR) has emerged as the biggest business idea to revamp the legacy processes as a reaction to the recession, where companies were forced to downsize their work force. Michael Hammer, who is also known as father of BPR, was the first one to effectively use and popularize the terms process improvement, process excellence and process innovation (Hindle, 2008). In his own words, BPR is envisioned as a “fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical measures of performance such as cost, quality, service and speed” (Hammer and Champy, 1993). The core idea is to motivate organizations to question the utility of their existing processes in terms of achieving their business goals. This in turn should lead to marked changes in organizational processes if found inefficient. One of the corner stone of the said approach is
intelligent incorporation of Information Technology (IT) in to business processes that leads to faster, cheaper, and better products (Hammer, 2010).

BPR is considered as a top-down approach, where the senior management take the steering role in an aggressive and autocratic leadership style to formulate and re-structure the processes with the help of IT tools (Carey and Chiarini, 2011). Balance should be achieved between decentralization of resources and centralization of processes through modern technology and equipment. BPR also results in replacing the hierarchal management structure to self-managing and self-controlling automated processes (Hammer, 1990).

One of the most quoted examples of process re-engineering is that of Ford Motor Company’s account payable process. They noticed that their Japanese competitors were closing down the gap in market share because of their faster production cycle and lower cost of production. On in depth analysis of their own processes compared with their competitors’ processes, they discovered many shortcomings. One of the issues was the slow and cumbersome working of their Accounts payable department that required extra manpower. This increased their cost as well as led to longer time to market for their products. Based on the idea of BPR, Ford made radical changes in their Accounts payable process (Graphically shown in Appendix C).

The old rule of "we pay when we get the invoice" was replaced by "we pay when we get the goods" and that too with the help of IT tools, databases and computer automation. This not only improved business process flow but also resulted in 75% reduction in headcounts. Moreover with the help of IT assisted automation, financial records became simpler and more accurate (Hammer, 1990). Muthu et al. (1999) discussed the importance of implementing BPR in a well-structured pattern and come up with following five steps for BPR inception in an organization:

(1) preparing for BPR;
(2) map and analyse As-Is process;
(3) design to-Be process;
(4) implementing reengineered processes with the help of IT; and
(5) improving continuously.

These steps are strikingly similar with the Lean tool of mapping the value stream. The underlying concept is to map the business activities in a way that eliminate the redundant activities or wastes in Lean’s terminology (Hammer and Champy, 1993). Although the goal of both BPR and Lean methodologies is similar, they differ in the way they are applied in organizations. BPR while streamlining the processes; is focused on activities from organizational point of view. While Lean suggests improving both the macro and micro activities with respect to employees day to day, work to attain an overall customer value. (Hines et al., 2004) BPR relies heavily on efficient use of IT tools whereas Lean considers the empowered people as its primary source. The combination of Lean and BPR approaches for process improvement will make sure the presence of IT as well as empowered people for eliminating wastes and NVA.
2.4 Theoretical Framework

Researchers hailed continuous process improvement as the key for organizations to improve their performance in terms of value they create for customers. Regardless of the wide acceptance of the benefits of process awareness and their subsequent improvement, it is still more *art* than *science* (Forster, 2006). Most of the existing literature on SPI focuses on plethora of competing methodologies, standards and practices yet making it very difficult to choose and apply one specific approach over others. Van Hilst and Fernandez (2011) argues that any discussion and implementation of alternative improvement approaches should reflect upon the underlying theories and assumptions. For given circumstances, these underlying theories frame and identify the problems and their relevant solutions, valid for that particular situation. In science, the underlying theories such as waves, relativity and quantum are well known and understood. On the contrary, for SPI, the underlying theories are still vague and incomplete (Forster, 2006). Therefore, the application of these theories is largely implicit (Van Hilst and Fernandez, 2010).

Harmon (2003) advocates identifying patterns in organizations to simplify complex circumstances and look for improvement opportunities. Van Hilst and Fernandez (2011) calls it a system of patterns and likewise any underlying theory, these patterns present “a framework of goals, solutions, assumptions that guide how we observe situation, define problems, and construct solutions”. Forster (2006) views the pattern systems as an approach to generalize, represent and organize process improvement constructs such that each pattern provides modification steps that can be applied to improve processes.

Most of the existing process patterns in software development deals with the processes and practices for *developing products* such as Ambler’s (1998) process patterns which give insights in to developing large scale systems using object technology. Ambler (1998) also referred *organizational patterns* in conjunction with the process patterns to be an important part of any improvement approach as organizational structures and management techniques play a critical role in it. Coplien and Harrison (2004) presented several of the organizational patterns especially for Agile software development. The top ten patterns according to him which form the foundations of any Agile organization are shown in Figure 5 (Coplien and Harrison, 2004):

![Figure 5: Coplien’s organizational patterns for Agile software development (Coplien and Harrison, 2004)](image-url)
Van Hilst and Fernandez (2010, 2011) come up with their system of patterns which is about processes and specific practices for improving processes in contrast with earlier discussed process patterns which focused on product development. Initially their proposed pattern model consists of five basic patterns for process improvement along with four anti-patterns to reflect critiques of process improvement. Figure 6 shows relationship between these patterns and anti-patterns for process improvement.

![Figure 6: Relationships among the 5 patterns (in blue boxes) and 4 anti-patterns (in orange boxes). Van Hilst and Fernandez's (2010) system of Patterns and Anti-patterns.](image)

The presented five patterns underlie several of today’s widely used process improvement practices (Van Hilst and Fernandez, 2010). Following illustration (Figure 7) shows key mapping of these patterns to established process improvement methodologies:

- **Have a Plan**: is emphasized in ISO 9000 standard for Quality Management as well as in PMI’s PMBOK (Project Management Body of Knowledge)
- **Copy What Works**: is emphasized in SEI’s (Software Engineering Institute) CMMI as well as in SPICE (Software Process Improvement and Capability Determination)
- **Eliminate Waste**: is the fundamental concept of Lean thinking
- **Consider All Factors**: is given due importance in TQM and Six Sigma.
- **Incorporate Feedback & Learning**: is the pivotal point in Agile methods and practices.

![Figure 7: Mapping of five patterns to existing process improvement frameworks (Van Hilst and Fernandez, 2010).](image)
Importantly, as a pre-requisite for any pattern model, each of these five patterns has a characteristic goal and corresponding solution. Regarding having a plan, Van Hilst and Fernandez (2010) describes its goals as to be prepared and consistent, while for its solution they argued about their availability at all times and making sure that they are known and followed by the employees. For copying what works, the goal is to achieve the same results as achieved by other organizations or industry leaders, which are being followed. The solution is to focus on the adoption of these best practices rather refining them. Achieving high efficiency is the main goal of eliminating wastes and this pattern can only be implemented by institutionalizing the commitment of waste identification and removal all across the organization. The goal for considering all factors is to achieve optimality in all aspects. The solution is to identify and analyse both internal as well as external factors that compromise quality and work towards fixing the root causes that gives way for later problems. Finally, for feedback and learning, the ultimate goal is to understand customer needs and delivering value added products and services to the customers. The solution is to shorten the feedback loop with customers, even involving them while taking their input frequently during development.

Similarly, Van Hilst and Fernandez (2010) also shed light on their presented anti-patterns about the mismatch between these anti-patterns’ goals and their respective adopted solutions. According to them ATAMO (And Then A Miracle Occurs) has a characteristic goal in sight but have poorly defined or missing solution process. In all problems are nails situation, there is a distinctive solution to an old problem but it may or may not fix the current problem. Buying a silver bullet demands a solution that is applied with little or no analysis of current problem. There is often a mismatch between the applied solution and current situation. Lastly the search to find a generic solution is also at times misguide people when there is an actual need for a specific solution to the situation at hands.

Van Hilst and Fernandez (2011) extends their research on the said topic and presented their system of patterns model for process improvement comprising of six underlying patterns as Plan, Best Practices, Flow, Feedback, Systems Thinking, and Living System. In their own words, they are of the opinion that “A process can be improved by having better plans (Plan), copying the most successful practices (Best Practices), reducing waste (Flow), using newer information (Feedback), addressing more factors (Systems Thinking), and having the process improve itself (Living System).”

Figure 8 shows the relationship among these six patterns for process improvement in an organization.
Likewise, in their previous model, all these six patterns are hailed from today’s best-known process improvement methodologies. Unlike their previous model where all the patterns appear to be converging on having and strengthening the Plan, in their new model the central focus is now shared between Plan and maintaining Flow. The addition of Living Systems pattern in their proposed pattern model is to emphasize the importance of a self-organized and self-governed atmosphere where employees can create their own order. They are motivated to suggest, decide and implement process improvement activities in a bottom-up way. Maintaining flow by eliminating wastes within the process along with adoption of living systems phenomena are fundamental principles of Lean thinking. Furthermore, Van Hilst and Fernandez (2011) iterates that these six patterns are not mutually exclusive and any process improvement activity following this model should apply all of these patterns in combination to get optimum results.

2.4.1 Derivation of Lean-BPR Adoption Pattern Model

The previous section detailed the difference between two approaches for pattern based improvement model i.e. practices based for developing products and for improving processes. As drafted earlier in the problem description and formulation section (section 0), this study intends to analyse ways to improve processes via incorporation of Lean and BPR rather than on developing products. The focus is specifically on the system of patterns models presented above for process improvement.

Both scientifically as well as practically, the impact of any change or improvement activity depends how it is measured. It is a well-known fact that, “if you can’t measure it, you can’t manage it” (Garwin, 1993). Most importantly, the selected metrics provides a way to gauge performances of different approaches under a similar situation. The main performance measure criteria for Van Hilst and Fernandez (2010, 2011) system of patterns is to observe how good the
plans are and how well they are followed. Thus making this approach quite subjective when it comes to measure the improvement performance. Over the years, several performance measurement systems have been developed for measuring improvements and one of them which is largely followed is named as “The Devil Quadrangle” (Forster, 2006). It suggests that Time, Quality, Cost and Flexibility are the basic performance measures for assessing all process improvement activities (Hayes and Wheelwright, 1988). Ideally, any process improvement activity reduces the execution time and cost, improves the quality of service and increase the ability to deal with variations (Duman et al., 2013). The interesting characteristic of this framework is while improving a process in one dimension, there may have a weakening effect on another dimension.

![The Devil's Quadrangle](image)

**Figure 9: The Devil's Quadrangle (Dumas et al., 2013)**

The authors of this thesis define the key performance indicators to measure the impact of their proposed Lean-BPR adoption pattern model based on these four competitive dimensions as depicted in the following figure (Figure 10):

![Key Performance Indicators](image)

**Figure 10: Key Performance Indicators for Lean-BPR adoption pattern model**

The authors of this thesis find several similarities between the patterns presented by Van Hilst and Fernandez (2010, 2011) and the Lean software development principles chalked out by
Poppendieck and Cusumano (2012). The following diagram Figure 11 shows the patterns from Van Hilst and Fernandez (2010, 2011), which are being used and/or translated to the proposed Lean-BPR adoption pattern model. It will be followed by a mapping table (Table 2) providing authors’ reasoning of selection or deselection of each of the pattern in Van Hilst and Fernandez’s (2010, 2011) models in the proposed Lean-BPR adoption pattern model in relation with the Lean software development principles.

![Diagram showing patterns adopted from Van Hilst and Fernandez's system of patterns]

Table 2: Derivation of Lean-BPR adoption pattern model

<table>
<thead>
<tr>
<th>Improvement patterns from System of patterns (Van Hilst and Fernandez, 2010, 2011)</th>
<th>Lean-BPR Adoption Pattern Model</th>
<th>Reasoning for in/out in the scope of Lean-BPR adoption pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider All Factors / Systems Thinking</td>
<td>Optimize the Whole</td>
<td>Lean thinking is based on creating value for customers and improving overall end-to-end processes.</td>
</tr>
<tr>
<td>Eliminate Waste / Flow</td>
<td>Eliminate Waste</td>
<td>Fundamental Lean principle of streamline processes and improving the flow of VA by identifying and eliminating wastes within the processes. Lean tools such as value stream mapping and BPR steps helps in waste identification and removal.</td>
</tr>
<tr>
<td>Incorporate Feedback and Learning / Feedback</td>
<td>Learn Constantly Keep Getting Better</td>
<td>Lean thinking emphasizes strongly on continuous improvement via shorter iteration of work units and frequent feedback loops to provide appropriate improvement opportunities in time.</td>
</tr>
</tbody>
</table>
| Living System | Engage Everyone  
| Build Quality In BPR | As a bottom-up approach, Lean advocates motivating all employees to participate in improvement movement. Like a living system, it promotes nurturing a culture where people who actually perform the work take the responsibility of improving their own work environment. This is also very closely knit with the BPR characteristics to use automation to improve quality in processes as a self-governing body. |
| Copy What Works / Best Practices | Not included | Instead of blindly copying solutions from external sources, Lean philosophy stresses on evolving process improvement practices which are rooted back to the current system and are foresighted by the persons while actually performing their day-to-day work. |
| Have a Plan / Plan | Not included | Performance measure criteria for Van Hilst and Fernandez’s (2010, 2011) system of patterns is to assess plans, whereas as discussed earlier in section (2.4.1), the proposed Lean-BPR adoption pattern model, measure improvements in form of KPIs based on The Devil Quadrangle. Lean thinking challenges the traditional approaches of making plans earlier to eliminate change during the process. In today’s dynamic business environment, preventing change is equated as being unresponsive to changing business conditions. Lean principles emphasize on the importance of interaction of people over processes, increased customer collaborations and responding to change over following a strict plan. |
| Anti-Patterns  
(Solutions must be General, All Problems are Nail, ATAMO, Buy a Silver Bullet) | Not included | The thesis framework focuses only on patterns for improvement and anti-patterns discussion is out of scope of this thesis study. |

Based on above discussion, the authors have derive the Lean-BPR adoption pattern model as illustrated in the Figure 12. The seven Lean software development principles are selected as process improvement patterns, which effects the KPIs of a software organization. Furthermore, four-step BPR process for incorporating automation as an effort for continuous improvement and
improving quality is also part of this model as a pattern itself. However the important notion here is that in the proposed pattern model the BPR steps are executed by the employees themselves i.e. working under a bottom-up Lean framework. The dotted lines among these patterns show their inter-relationship e.g. the three patterns eliminate waste, keep getting better and build quality in, are closely associated with the overall BPR implementation effort. Likewise Van Hilst and Fernandez’s (2010, 2011) system of patterns, the patterns in the adopted model are not mutually exclusive and should be applied in combination to achieve desired improvements in KPIs. This Lean-BPR adoption pattern model will be the reference framework for the analysis and generalization of this thesis via a case study.

![Figure 12: Derived Lean-BPR adoption pattern model](image-url)

2.4.1.1 Thesis hypothesis

Based on the above-mentioned thesis framework, the authors hypothesize that:

Adopting Lean software development principles and incorporating BPR steps bottom-up, will improve the Key Performance Indicators (KPIs) for a software development organization in terms of:

- cost reduction
- increased productivity
- improved quality of products
- and better resource management
3  Method and Material

This chapter describes the selected research methodology for conducting this thesis project along with the alternatives considered during the course of this research study. It will detail the in-depth description of method decisions to get the relevant data for analysis. Furthermore, this chapter ends with the discussion on the reliability and validity of results.

3.1 Stages for thesis project

The study during this thesis project can be categorized into two stages, i.e. literature review stage and a case study based on the theoretical framework which is adopted from the reviewed literature. In literature review stage, significant publications, books and internet articles are critically examined for existing and potential SPI methodologies including Lean and BPR. The literature provided the theoretical foundation for the case study conducted for this Master thesis report. The defined purpose for this study is to analyze if and how Lean thinking and principles can be used for improving software processes as well as how BPR, which is primarily a top-down approach can be implemented under a bottom-up Lean framework. Yin (2013) favors case study approach when the focus of study is to answer if and how scenarios (Yin, 2013).

3.2 Research methodology selection

For scientific purposes there are two popular approaches used to answer specific research questions i.e. qualitative and quantitative methods. Qualitative research helps in understanding the narrative and descriptive meaning of a particular phenomenon for a certain group of people. In other words, it helps in developing a broader picture based on information received from individual participants (Guest et al., 2013). Quantitative methods on the other hand, aim at explaining phenomenon through collection and analysis of numerical data (Creswell, 1994). The common techniques used for qualitative research are observations and interviews whereas quantitative research is often based on laboratory experiments and numerical data collected through other methods (Merriam, 2009).

This research study incorporates both Qualitative and Quantitative methods and data to analyze the objectives of this study.

Qualitative approach was used to assess following key issues.
- To understand areas in the software development life cycle that needed improvement from an overall business perspective.
- To assess understanding of lean principles within the organization.
- To get employee and management perspective on the idea of combining Lean and BPR approaches.

This was facilitated by using semi-structured questionnaire for interviews. The semi-structured questionnaires enable interviewers to conduct a dialogue with interviewee and to obtain clarifications where needed (Denscombe, 2000). The questionnaire was sent to selected participants through the company’s in-house survey tool prior to interviews.
Quantitative approach was used to test the hypothesis i.e. improvement in KPI’s through the newly adapted Lean and BPR based changes in the legacy process of software development.

Combining the two research approaches, i.e. “Mixed-Method Research” is a well-known and established entity in academic settings (Johnson and Christensen, 2010). The use of mixed method approach is potentially capable of providing a better understanding of problem as well as of combining the advantages of the two methodologies. It also enables researchers to look at the problem from multiple viewpoints thus adding accuracy to their judgments (Todd 1979).

For the case study, Motorola Solutions has been selected as an organization as the authors of this thesis, being its employees; have access to the data as well as the capacity to analyze and re-engineer the software processes. The methodological framework of Plan-Do-Check-Act (PDCA) was used to implement Lean-BPR adoption pattern model’s patterns in the software development life cycle for the case study organization. This framework was developed by Deming (1950) and has been used to implement change and improvement in management systems (Carey and Chiarini, 2011). Researchers like Dennis and Shock (2007) have evaluated PDCA as methodology for implementing Lean. Similarly Muthu et al. (1999) devised PDCA based steps for practicing BPR in organization, thus providing basis for its use in this project.

In the Plan stage, objectives for process restructuring, strategies for implementation as well as required resources in terms of manpower and equipment were identified. In the next stage (Do), the planned activities were actually implemented as part of the case study. Deming does not advocate use of any particular tool for Do phase rather it should be chosen according to the requirements of organization. In our case, IT-assisted tools such as IXIA as well as value stream mapping were used for implementing Lean and BPR principles for process improvement. The outcome of these activities i.e. re-engineered process was analyzed and revised for continuous improvement in Check and Act phase.

One of the alternative considered during this study was to use Intervention or Laboratory approach for the case study. In such studies, two groups are formed, one which is called the experimental or intervention group and the other is control group. In the former group, changes or improvement actions are taken while there is no change in the way of working in the later group. And at the end of the study, measured results are compared with each other to analyzed the effects of changes or improvements. Such approach are difficult to arrange as well as requires time and extra resources to run these pilot projects. This approach is extensively used in pharmaceutical industry for clinical trials of different drugs but practically, in software development organization to divide employees e.g. engineers, testers etc. in to two groups performing the same tasks or developing the same software in different ways can be demotivating. Furthermore as the authors’ already had the data from the past when Lean-BPR adoption pattern model is not used, they can used this data to compare the case study results after implementing Lean-BPR adoption pattern model.

3.3 Data collection strategy
For this case study, data collection came from multiple independent sources (Bryman 2006) i.e. surveys, interviews and archival data/documentation etc.
3.3.1 Archival data collection strategy
Data for quantitative analysis was collected using change management software called IBM Rational Clear Quest (CQCM). This tool provided information on the quality of product such as number of defects found in the software in relation to the time of their detection and rectification. Time and resource management information was calculated by Primavera project management tool. This information was used to calculate savings in terms of time and cost through implementation of restructured test process.

3.3.2 Interview strategy
This adopted Mix-Method approach was facilitated by using semi-structured questionnaire for survey. The semi-structured questionnaires enable interviewers to conduct a dialogue with interviewee and to obtain clarifications where needed (Denscombe, 2000). The questionnaire was sent to selected participants through the company’s in-house survey tool prior to interviews. This survey was sent to individuals of different departments that have an impact on the overall value stream and are involved in end-to-end software development lifecycle. The questions were kept open-ended to give the recipients the flexibility to assess and answer according to their own daily work routine and observations. Most importantly, the questions were designed to reflect participants’ understanding of the patterns of Lean-BPR adoption pattern model. The survey was followed up with informal interviews or dialog sessions with the individual teams first and then with all the teams together.

Interviews were conducted face to face or through telephone depending upon geographical location of participants. In total, survey was sent to 21 recipients belonging to five departments namely Program Management, Finance, Customer Relation Management (CRM), Software Development, Test and Quality Assurance departments. Informal interview sessions were conducted to follow-up on answers given in the survey with individual teams first and then a combined session with presence of all the departments.

3.4 Reliability
The answers to survey questions are not necessarily always same if questions are repeated over time. The answers are reflection of interviewee’s perception and knowledge at that certain time and can vary due to changed circumstances. Similarly, to know at what time the measured data is collected is also a very important factor in reliability discussion. For this thesis project, the survey was done before any implementation of Lean-BPR adoption pattern model and the main aim was to know the current in-practice software processes and challenges employees were facing at that time. Previous data in terms of identified KPIs is also gathered at that time which should then be compared with the results after adopting Lean-BPR adoption pattern model. The implementation based on the proposed framework was started in March 2011. To get reliable data, data collection did not start until three months later in July 2011 in order to wait for processes to stabilize and adapt to the changes as per Lean-BPR adoption pattern model.

Another important aspect while comparing results with previous data is to know about the complexity of work among these different software development releases. This has been verified with the project managers to keep the same number of software features developed in each
software release for data collection purpose. This enables the authors of this thesis to maintain balance with respect to workload while comparing results acquired at different timeframe.

Another important aspect of reliability in research is also about gauging the reliability of interviewees as well. In other words, it should be possible to verify the identity of interviewees as well as it is important to know that the persons themselves produce the answers. This has been ensured by the online survey tool as well as talking with the interviewers directly face-to-face or by telephonic conference call.

3.4.1 Form of bias

There are several factors that can enhance the bias of the interviewee such as level of knowledge of the researcher, information provided to the interviewees, nature of questions and physical location etc. In order to minimize the information bias in interview responses, a three-stage interview strategy as discussed earlier is adapted. Firstly an online questionnaire along with relevant information about the subject matter has been sent to interview participants. Secondly, informal interview sessions with the individual teams are conducted and lastly a joint session of all the participating teams is carried out.

For minimizing selection bias, the participants for the survey were selected based on their respective roles in the departments and understanding of the end-to-end software development life cycle and included both managers and experienced employees.

In order to not having any influence of authors’ knowledge about the subject matter, the effort has been made to keep questions neutral and open ended. As authors are part of the organization themselves, it helped in developing the relevant questions which give insightful information about organization’s processes and areas for improvement. However, before sending these questions to participants, these questions were verified by testing them on authors’ own team and the received feedback was used to re-formulate questions with more clarity. And also as mentioned above the three-stage interview strategy also helped in eliminating the doubts about the purpose of questions among participants especially during informal discussion sessions. Furthermore, data collection approach from multiple sources (in form of discussed IT tools) helps in eliminating the internal data bias.

3.4.2 Validity and generalization

This section discuss the validity concerns about adaption and translation of the Lean-BPR adoption pattern model in to the reality by conducting a case study. Researchers like Yin (2003), categorized validity in to three categories, which are discussed below:

3.4.2.1 Construct validity

Construct validity deals with selecting appropriate type of changes to be studied and then relating them back to the research objectives (Yin, 2003). In this study, all the interview questions are associated to each of the pattern of the Lean-BPR adoption pattern model.

3.4.2.2 Internal validity

Internal validity emphasizes on matching the findings or outcomes from the case study to the reference model by minimizing or completely omitting the extraneous factors from the study. For
that, the results of the case study are analyzed by mapping them to hypothesis (section 2.4.1.1) derived from the theoretical framework of this thesis.

The parallel consistency check was also performed by adopting three stage interview processes. This help the authors to minimize the inconsistencies between the data gathered during the interviews from different stakeholders who have different roles/perspective on the subject. Moreover, member check as part of internal validity was also practiced by sending participants a draft copy of data analysis for their valuable feedback.

3.4.2.3 External validity
External Validity refers to the generalizability of the research findings to the outside world. It actually questions if same findings or result can be achieved under similar conditions at other places Yin (2003). The issues identified during the course of this case study as well as the patterns of the Lean-BPR adoption pattern model are then matched with other software development companies’ problems and their remedy actions by reviewing other relevant case studies. In this regard, authors have made an effort to provide detail description of steps taken to implement Lean-BPR adoption pattern model in this case study so that they can be repeated at other external places.
4 Case Study

This chapter accounts the details regarding the case study of a software organization, where process improvement activities are carried out with reference to the thesis’s theoretical framework presented earlier in theory chapter. The initial sections provide a brief introduction of the software company, software development model it uses as well as some apparent problems it is facing with respect to its processes. This is followed by summary of results obtained from surveys and interviews of participants which helped in understanding company’s current software development process. These findings are analyzed and various process improvement actions are implemented under the guidance of the Lean-BPR adoption pattern model. The empirical results of the implemented process improvement actions are listed afterwards. This chapter concludes with a summary of issues found, their mapping with the Lean-BPR adoption pattern model and the tools used for improvement respectively.

4.1 Organization’s Background

Motorola Solutions is a telecommunication organization providing software and hardware products and services to both enterprise and government sectors. In the enterprise segment, its products include barcode scanners and two-way radio etc. for a variety of worldwide customers. In the government segment, it provides end-to-end services for mission critical public safety communication for customers like police, fire and ambulance etc. According to Motorola Solutions Annual Report (2011), the net sales for the year 2011 were $8.2 billion, out of which government segment contributed 65% while 35% came from the enterprise segment. Due to the 2/3rd contribution in the net sales, the government segment plays a vital role in the successful business of the company.

4.2 Organization’s Product Development Model & Apparent Challenges

Motorola Solutions Engineering department for its government segment comprises of many teams which are responsible for designing and developing their respective deliveries which are then integrated together to launch the final mission critical public safety product. Motorola Solutions has developed its own product development model called Motorola Gates (M-Gates) which is inspired from the Waterfall software development life cycle model. M-Gates is a common framework across Motorola Solutions that ties together cross-functional teams participating in all system development phases. The 16 M-Gates represent a comprehensive set of Marketing, Engineering, Project Management, and Manufacturing activities that will ensure proper business planning, timely business decision and execution. As shown in Figure 13, it is the timeline of several checkpoints/gates where project progress is monitored and key decisions are made before proceeding to next gate.
Motorola Solutions’ Quality department has reported an increase in defects found during various stages of product development life cycle (M-Gates). This rising number of defects is partly explained by rapid increase in its customer base as well as its feature set. With increasing complexity of Motorola Solution's product features as well as diverse needs of its customers, Motorola Solutions' high quality standards could only be maintained through revised and improved processes that ensure better test coverage and will reduce the number of defects passed on to customers.

In financial terms, impact of defect detection and resolution depends largely on product development lifecycle stage i.e. M-gates at which the defect is identified. Figure 14 summarized the effect in terms of cost and time to repair a single defect found in various M-gates. A resolution of a single defect detected in earlier stages of development i.e. in M7-M5 cost on average around $5000 and around 2 weeks of staff time. A slightly delayed detection at implementation phase i.e. M5-M4 increases the cost and resolution time by three folds. These figures rise exponentially if the defect is detected after delivery to the customer. In the later case, a Failure Review Board (FRB) examines the defect and probe into the causes of that defect. A new service pack (SP) needs to be rolled out to all the other effected customers using that product. This leads to NVA management tasks at both program and operational levels.
4.3 Results of Survey and Informal Interviews session

Based on Lean-BPR adoption pattern model, the major aim of this activity was to engage all the concerned person in the software development lifecycle in order to understand the current way of doing things, assessing them with the people who actually perform the work and seek suggestions for the areas of improvements. Secondly, the said activities were also designed in a way to understand participants understanding of Lean thinking and how they associate it with software development. Lastly, participants were presented with the concept of combining Lean and BPR for software development and getting their perspective on that.

Following sections present, the summarized results from the feedback received by both activities i.e. survey and follow-up dialog sessions.

4.3.1 Improvement areas in current software development process:

Question: According to your experience what are the problems in the Motorola Solution’s software development lifecycle that needs immediate improvement? (List at least 3 areas)

Results:

All participants answered that question in the survey with the list of problem that they think, should be improved. The answers to this question were further discussed in the interview sessions. The problems repeatedly mentioned by the participants could be categorized as under:

Workload: Test team members repeatedly complained about the excessive workload. It was further explored in the discussions that the increased work load for down stream test teams was mainly due to the fact that upstream development teams were working in short iterations.
following the Agile way which was not synchronized with the test teams. Different teams mentioned testing as a bottleneck in the entire software development process flow.

**Time to correct errors:** Development teams were concerned about delayed detection and notification of defect. As the defects are often reported late, there is always a possibility that these residing bugs might have introduced new errors and that eventually require more time to fix those issues.

**Task switching:** This issue is also linked to the late detection of defects during the software development lifecycle. There was often a scenario where development teams needed to prioritize if they continue the development they are doing or fix the bugs reported late by the test teams.

Project managers highlighted that they are facing difficulties to manage limited resources during the software project time frame. During the discussion they elaborated that, once the product is at post implementation stage i.e. Gate-M2, assigned expert resources are partially moved to other customer projects and assignments. When a defect is found after implementation phase (post M3), Subject Matter Experts (SME) are called back from their new assignments for defect resolution. From the SME’s perspective, this task switching was reported as a source of distraction. This also led to delayed completion of both new and old tasks on their behalf.

**Poor Coordination:** It has been observed that there is better coordination among the upstream teams including the business, product marketing and the software development team. They have daily stand up meeting for discussing the status of work. On the other hand the downstream teams including System Integration Test (SIT), Customer Acceptance Test (CAT) and deployment team are not in the overall loop of communication. They often feel as neglected earlier but becoming more important during the later stages (in Gate-M3 etc.).

**Test Coverage:** The participants from the Quality department highlighted that the test coverage is not proportional to the amount of new features developed. During the combined discussion involving all the teams it was discussed that one of the reason could be the long time needed to conduct manual testing. This has a direct effect on work load for the testers, resulting in decrease productivity over time and in overall quality of the product due to untested segment of codes in the delivered product.

**Question:** *In your day-to-day work, what metrics do you use to measure the performance?*

**Results:**

This question was asked in order to understand both the employees and management perspective of KPIs while developing software. All participants answered that question in the survey with the list of performance measurement matrices they consider while working. Some of the common ones are:

- Meeting project deadlines.
- Number of customer complaints.
- Number of new features developed and defects fixed per iteration.
• Number of test cases executed (Test coverage of specific features/Customer requirements).
• Time to find a defect.
• Time to correct a defect.

All the above-mentioned metrics are useful in measuring performance in terms of cost, productivity and quality of the final product and are aligned with the KPIs mentioned earlier in section 2.4.1 as part of Lean-BPR adoption pattern model.

4.3.2 Understanding and practice of Lean within organization:

Question: While at Motorola Solutions, what experience you had with Lean so far?

Results:

This question was responded with mixed answers that can be grouped depending on participants’ department affiliations. The management and business teams mentioned about the Lean training provided to employees and the promotion of “quick wins” (see Appendix D) as part of organizational effort to become a Lean company. However, the training material so far available for employees is of general Lean principles and not specifically tailored to software development. Upstream teams, especially the development teams are of the opinion that they are already practicing Agile (Scrum) way of development and that itself means that they are following Lean. This misconception relating to lean and agile as reported by software development teams is in line with what has been reported previously in literature and discussed in earlier section (section 2.2.1). Downstream team including the CAT and deployment teams etc. mostly mentioned Lean practices only for Motorola Solutions’ manufacturing or supply chain businesses.

The answers to this vary question clearly reflects the way the Motorola Solutions employees are trained so far in Lean philosophy. They have the general understanding but lack insights of Lean software development principles. This project served as a catalyst in promoting the Lean software development principles as patterns of Lean-BPR adoption model across the organization. Both the higher management and the employees become aware of importance and area of impact of Lean-BPR adoption model through the conducted informal dialog sessions.

4.3.3 Combining Lean and BPR approaches:

As a pre-requisite, the participants were briefly informed about the bottom up Lean and top down BPR approaches before the following survey question.

Question: How do you see Lean and BPR working together?

Results:

The questionnaire result on the said question does not give any useful information as most of the participants opt to skip that question. However, during informal interview sessions, some constructive discussion took place with regard to Lean and BPR co-existence. Here are some of the important points discussed:
The concept of BPR was very popular in 1990s and subsequently it is used in combination with other process improvement approaches.

Lean tool of value stream mapping resembles closely with BPR steps.

Management reaction in case of employees themselves deciding for automation and IT tools while adopting top-down BPR steps.

Finally yet importantly, employees shared their concerns regarding BPR approach. It was a general belief that implementing BPR can lead to large cut downs in organizations as IT tools and automation can take over lots of jobs, as observed and reported frequently in late 1990s.

4.4 Implementation details of Lean-BPR adoption model

The results from the questionnaire and informal interviews with the employees from various upstream and downstream teams helped in understanding and assessing the current situation and areas of improvement within the company processes. Regarding poor quality of products, most of the concerns are raised regarding the test process and its relationship within overall software development life cycle. The patterns of Lean-BPR adoption model such as eliminating waste and specifically BPR pattern steps emphasize on improvement opportunities by first identifying the process to work on and then look for waste removals from it. In current scenario, SIT process plays a critical role in the software project life cycle taking the majority of burden for testing. Figure 15 maps the Test process work flow among the development, SIT and CAT teams in form of a sequence diagram.

Once the product requirements are locked down at M-Gate 8, various software teams proceed with actual development and internal/unit testing of their software components. Instead of waiting for one final product, its different components (features) are delivered to downstream SIT team in four iterations named as loads. Functional Loads (FL1 and FL2) comprises of product features developed by different teams, which are delivered to SIT for integration and system level testing. Maintenance Loads (ML1 and ML2) contain revised product features in light of defects reported by internal testing as well as by SIT.

Traditionally SIT took 25 days to create a test strategy and define/write relevant manual test cases for all the planned features in a complex external test management tool as a first step. Prior to each load delivery to CAT team, SIT took 5 days to conduct tedious manual testing which required installing different network configurations to a large number of network equipments such as routers, switches and firewalls. Identified defects by SIT were reported in the IBM Rational Clear Quest software which were then analyzed and fixed by the relevant development teams and re-tested by SIT to ensure conformance to requirements in subsequent loads (iterations).
Figure 15: Legacy SIT test process workflow (As-Is state)
As part of analysis, the current SIT test process flow is examined in a way to find NVA tasks that match the patterns of Lean-BPR adoption pattern model. Once these NVA are linked to patterns of Lean-BPR adoption pattern model, respective tools and activities can be used to eliminate these NVA and improve the SIT test process.

With successive loads, SIT not only has to test the new features that were introduced in that particular load but also re-test the previous load features to ensure their compatibility with new features. The re-testing however meant repeating the cumbersome manual test cases that result in loss of both time and cost for SIT. Furthermore there was an increased chance of human errors. The repetitive work not only cost valuable time of skilled resources but also led to boredom among team members. SIT were also over burdened by the fact that work load increased with each successive load but the allocated time of performing the testing task was unchanged i.e. 5 days. Testers were required to develop manual test cases at the start of first iteration for all the iterations is clearly overloading the resources with extra/unneeded tasks that might not be valuable later in the software development lifecycle. All the above examples can be categorized under NVA and can be mapped to various wastes form as discussed in section 2.2.2.2, such as overburden, handoffs and extra/unneeded processes etc.

The Quality department was also extremely concerned on increasing number of defects being passed from SIT team to CAT and/or field test teams and indicated a need to increase the test coverage. With only manual testing it was not possible to cover wide range of software testing scenarios because of technical as well as time constraints. The pattern build quality in as well as BPR steps actually address this issue of automating the testing especially the regression tests that are supposed to be run multiple times and manually running them will not only introduce human errors but they also consume much time.

The authors of this thesis came up with a Plan\textsuperscript{1} to re-engineer the SIT test process by obliterating unnecessary steps in testing process and using IT assisted automation by incorporating appropriate tools as prescribed by patterns of Lean-BPR adoption pattern model.

As a first step, Lean tool of value stream mapping was used to create and analyze the current state of SIT test process and to propose future state of re-engineered test process. This mapping enabled visualization of VA and NVA activities, the two frequently discussed themes in Lean thinking (Mujtaba et al., 2010) (Mary and Tom Poppendieck, 2003).

Figure 16 and Figure 17 represents the current and future value stream mapping of SIT test process for FL1 period respectively. Similar value stream maps were created for all 4 iterations.

---

\textsuperscript{1} Deming's Plan Do Check Act (PDCA)
Figure 16: Current value stream map of SIT test process at FL1 time frame

Figure 17: Future state value stream map of proposed SIT test process at FL1 time frame
As a confidence building measure, test team (SIT) employees were asked to look for some quick wins (see Appendix D) in the area of their day-to-day work. In Lean terminology, it is equivalent to a Kaizen events i.e. achieving small and focused process improvement in an accelerated manner. It is an important Lean tool which encompasses multiple patterns of Lean-BPR adoption pattern model such as learn constantly, keep getting better and eliminating waste. They come up with the following major quick wins which are identified and implemented quickly at the start of a Do\(^1\) phase paving the way towards the desired future state as planned:

- SIT has an expensive hardware tool (IXIA) that costs around $100,000. Despite of its extensive feature sets for which Motorola Solutions is paying licensed fees as well, it was minimally utilized in the testing process i.e. only for generating dummy network traffic. It was identified that this tool can be used to run automated test cases by running tests scripts on it. As a first step couple of manual test cases were implemented at IXIA via test scripting as a pilot project. This led to a quick win in just a week and give confidence to the team that the tool can be used for extensive automated testing.

- SIT and CAT/field test teams were using a common yet complex Test management tool. SIT only used this tool to define test cases and align them with the requirements. After test cases execution, outcome and results were also saved in it. The application's complex features served well for CAT/field test team's requirements but were too complicated for SIT. It took additional efforts in terms of coordination and communication with downstream teams as well as extra time to fill out details which were not even relevant for SIT. As an attempt to remove these NVA steps in test management, SIT evaluated various options. It was identified that acquiring an off the shelf tool will require long time and also high licensing cost in purchasing. The team therefore developed a customized tool in MS Excel in a week time that served the purpose. Obliteration of this NVA work as well as incorporation of suitable IT tools to increase efficiency is what Hammer (1990) advocated for BPR.

- The health status of software systems in general as well as specifically after test cases execution was checked by physically visiting network lab and obtaining and recording information manually. This process at times become hectic because of distant location of network lab from the testers work desks as well as gathering data from a large number of network equipment such as routers, switches and firewalls. Shingo (1981) classified the above situation as the unnecessary motion that requires time and effort from employee later to reestablish focus of what he/she was doing before moving. SIT developed set of automated scripts that can be run to obtain network health status while sitting at their desks. This was a quick win achieved in one day and become an instant hit across other development teams.

These quick wins are linked with the organization’s reward and recognition program and respective employees were awarded “High Five” awards and monitory rewards for their

---

1 Deming's Plan Do Check Act (PDCA)
successful work and their success stories were also displayed in televisions mounted in coffee areas. All these quick wins helped in moving a step further towards attaining the bigger goal of improving the test process by adoption of Lean patterns among employees and integration of IT tools to efficiently restructure the process flow i.e. a basic BPR philosophy. The idea was to replace the old SIT rule of "We manually test what is planned at the project initiation" with "We test efficiently what is to be delivered to the customer (CAT) at a particular time utilizing automation".

In the restructured test process, initial 25 days period for SIT test team was replaced by shorter period of 5 days before each load during which test cases specific to that particular load were designed. It resulted in improved communication with the upstream team to decide and plan for load specific test cases.

It was beneficial in multiple ways, as it not only saved time but also reduced over-burden among team members. The initial 25 days chunk of work was a NVA task for SIT as it led to over production of test cases that were not required at that very moment and sometimes were not even used later. Previously SIT work was dependent on what was pushed by the development team. With the new process, SIT was actively involved in feature selection for specific loads and thus was able to establish pull on the development team to deliver the required and pre-decided features for testing. This improvement action of establishing the pull is linked specifically to the deliver fast pattern of the Lean-BPR adoption pattern model as discussed earlier in theory chapter. Furthermore, this division of work in to 5 days chunks for each load also removed variability from work cycles as they are now of equal duration. As a result, resource management was improved in this re-engineered process, as the tasks were now well defined and spread over shorter and fixed duration of time. This also helps in efficient management of specialized test resources depending upon the priority of the feature and availability of lab.

Test case management was also improved in the restructured process. In the past a lengthy test case document was developed early in the project. Reviewing this large document was complex and time consuming for reviewers as they have to look at all developed test cases in one go. This was replaced by four load specific test case sub-documents generated by MS excel tool (developed in-house as a quick win), which were reviewed before each load respectively. Reviewers were now able to focus more as they were provided with the required information at the correct time. Numbers from Formal Technical Review (FTR) tool indicated that with restructured test case documentation, 30% more comments were received from the reviewers than in past. This improved feedback was highly valuable for SIT and enabled them to define appropriate and precise test cases that led to improved test coverage.

Successful use of IXIA tool for automating test cases in the pilot project was followed by the decision to replace entire manual testing with automated testing. In manual testing, more time was consumed in execution with very little time to analyze the test results. Analysis was carried out in the short time frame in between the manual execution of two test cases. Moreover the analysis was very subjective to the test executer and interpretation differed between two individuals. With automation the time distribution was reversed, with very quick automated execution of all the test cases and ample time available for analysis by the test experts. As a
byproduct of this effort, working conditions for tester also improved as they no longer had to stay in noisy lab for 5 days manual testing. Broadbent and Little (1960) as early as in 1960 identified that noise reduction at work place leads to increased efficiency and less human errors.

In the **Check and Act**\(^1\) phase of test process restructuring, *lesson learned* sessions were carried out after each load rather than one at the project closure. *Feedback* from each session was utilized in the successive load, forming the basis for continuous improvement in the process. These repeated sessions were more productive because team members could easily remember what they have done in the recent past and what improvement could have been made to attain better outcome and increase productivity. Previously when the feedback was obtained at a very late stage, it was hardly possible to make any improvement in the current work based on that feedback.

Figure 18 shows the sequence diagram of the SIT restructured test process flow:

\(^1\) Deming’s Plan Do Check Act
Figure 18: Re-Structured SIT test process workflow
4.5 Empirical results obtained after implementing Lean-BPR adoption model

Each year Motorola Solutions make one major software delivery to its customers. The process improvement initiative under discussion was carried out for D8.0 release. The results from this release were then compared with past three releases (D7.0, D7.1 and D7.2). The authors of this thesis have quantified the major improvements after test process restructuring. This helps in comparing different KPIs like test coverage, cost and time etc. with previous releases.

Figure 19 shows number of test cases conducted by SIT over four successive releases. In D8.0 with automated testing, test coverage improved radically by 79% compared to last release as SIT was able to execute much larger number of test cases.

![Figure 19: Conducted Test Cases per releases](image)

Improved test coverage in D8.0, resulted in increased early detection of defects by SIT. An incremental increase in number of defects detected by SIT in each successive load of D8.0 can be seen in Figure 20.

![Figure 20: Defect Detection in each releases](image)

Saving in terms of time is shown in Figure 21. Duration for SIT testing effort for each load reduced from 5 days to 2 days i.e. 60% reduction from previous releases.
Time was not only saved in quantitative terms but as discussed earlier it was now spent more on value added task of test analysis rather than NVA of manual test execution (Figure 22).

Figure 22: Time spent on Test cases Execution vs. Tests result Analysis

Figure 23 reflects on one of the major aims of this case study project i.e. to decrease the number of defects passed to the customer (in this case as CAT team).

Figure 23: Defects raised (by downstream) per load per release
Referring back to the Motorola Solutions' estimated numbers (shown in Figure 14); reduction in number of defects found by CAT resulted in reduced cost and time needed to fix those defects. As shown in the following Figure 24, estimated cost of fixing defects raised by CAT team in D8.0 is 600000$ as compared to 1350000$ in D7.2, which shows the estimated reduction of 44%. Similarly estimated man weeks for defect fixing is reduced from 324 (D7.2) to 144 in D8.0.

![Figure 24: Estimated cost in terms of $ and man weeks](image)

4.6 Important Findings of the Case Study

Table 3 summarizes the issues identified during the analysis of case study along with potential risks for software organizations. The organizational risks are identified from the literature review presented in section 0. Furthermore, these findings are mapped to the patterns of the Lean-BPR adoption pattern model and the table also lists the tools and/or actions taken to address these issues.

Note: The established tools for Lean software development as presented by Mary and Tom Poppendeick (2003) are listed in Appendix B)
<table>
<thead>
<tr>
<th>Issues / Findings</th>
<th>Risks for organizations</th>
<th>Patterns (of Lean-BPR adoption pattern model)</th>
<th>Tools and/or Actions taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding of Lean patterns for software development</td>
<td>Lack of understanding of customer value and employee empowerment.</td>
<td>• Engage Everyone</td>
<td>Improve organizational Communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Engage Everyone</td>
<td>Leadership support in promoting Lean culture across whole organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Learn Constantly</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Keep getting Better</td>
<td></td>
</tr>
<tr>
<td>Change of organizational culture and human behavior</td>
<td>Lack of motivation &amp; resistance to change among employees.</td>
<td></td>
<td>Empower employees by motivating them and involving them in decision making process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Initiate Rewards and Recognition program for employees accomplishments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Improve Feedback and organizational Communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Promote Kaizen improvement events (Quick wins)</td>
</tr>
<tr>
<td>Extra management tasks</td>
<td>Project delay, Source of wastes such as delay, handoffs etc.</td>
<td>• Eliminate Waste</td>
<td>Value stream mapping tool and 4-step BPR process to identify and eliminate wastes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Build Quality In</td>
<td></td>
</tr>
<tr>
<td>Frequent changes in requirements and/or customer's needs</td>
<td>Project delay resulting in increased cost, increased risk of defects, demotivating for employees.</td>
<td>• Deliver Fast</td>
<td>Incorporate Options Thinking i.e. decide as late as possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Learn Constantly</td>
<td>Establish Pull systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Working in Iterations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Feedback</td>
</tr>
<tr>
<td>Problem Area</td>
<td>Issue Description</td>
<td>Measures</td>
<td>Recommendations</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Over Production</td>
<td>Waste of employees’ time, Increased cost, NVA tasks with regards to customer’s need</td>
<td>• Eliminate Waste</td>
<td>Value stream mapping tool and 4-step BPR process to identify and eliminate wastes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Deliver Fast</td>
<td>Establish Pull systems</td>
</tr>
<tr>
<td>Lack of overall picture or value stream for employees</td>
<td>Misunderstanding of organizational goals, difficulty in transforming organizational goals in to daily work life, lack of knowledge with regards to customer value</td>
<td>• Optimize the Whole</td>
<td>Value stream mapping of all processes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Engage Everyone</td>
<td>Improve Feedback and organizational Communication</td>
</tr>
<tr>
<td>Lack of coordination among various teams</td>
<td>Lack visibility into other teams and projects, lack of trust among employees, project delay, increased risk of defected product</td>
<td>• Engage Everyone</td>
<td>Establish Pull systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Working in Iterations</td>
</tr>
<tr>
<td>Suboptimization</td>
<td>Increased risk of localized improvement only e.g. different teams attempt to reach a solution that is optimal for that particular team’s tasks, but that may not be optimum for the organization as a whole.</td>
<td>• Keep getting Better</td>
<td>Define Metrics for measuring performance &amp; improvement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Optimize the Whole</td>
<td>Raise employees motivation and involve them in the decision making process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Engage Everyone</td>
<td>Use the expertise of the employees who perform actual work</td>
</tr>
<tr>
<td>Late detection of errors &amp; long time to correct those errors</td>
<td>Project delay, increased cost to project, effect on resource planning and management especially of developers and testers.</td>
<td>• Deliver Fast</td>
<td>Establish Pull systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Build Quality In</td>
<td>Automated Testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Automation (BPR)</td>
<td></td>
</tr>
<tr>
<td>Low test coverage</td>
<td>Decrease productivity and quality over time due to untested segment of codes in</td>
<td>• Build Quality In</td>
<td>Automated Testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Automation (BPR)</td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td>Solution</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Repetitive tasks for employees</td>
<td>Increased risk of human errors, unchallenging tasks creates boredom, lack of</td>
<td>Automation (BPR) 4-step BPR process to identify NVA and incorporate automation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>motivation and encouragement among employees.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cumbersome manual paper work</td>
<td>NVA and source of wastes such as delays, waiting etc. loss of tacit knowledge in</td>
<td>Value stream mapping tool and 4-step BPR process to identify and eliminate wastes</td>
<td></td>
</tr>
<tr>
<td>Handoffs</td>
<td>between handoffs, longer time to market.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Switching</td>
<td>Uncertainty and distraction among employees, increased project delivery time as</td>
<td>Value stream mapping tool and 4-step BPR process to identify and eliminate wastes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>employees need to refocus/relearn the old project details after moving to new project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overburden resources</td>
<td>Project delay, increased risk of inventory work and human errors, increased response waiting time.</td>
<td>Establish Pull systems Value stream mapping tool and 4-step BPR process to identify and eliminate wastes</td>
<td></td>
</tr>
</tbody>
</table>
5 Discussion

This chapter discusses the findings of the thesis case study along with its analysis and relate them to the existing literature and research in the process improvement domain. All the discussion is founded on the principle of moving closer to the theory by generalizing and externally validating thesis results with the existing research. In this process, differences and/or disparities between this study’s results and outside world are discussed for each dimensions/patterns of the Lean-BPR adoption pattern model.

As discussed earlier in theoretical framework section (section 2.4.1), individual patterns of the Lean-BPR adoption pattern model are not mutually exclusive and in between themselves, they may share similar tools for improving a particular dimension of the model. Similarly, the individual issues and/or findings of the thesis case study as presented in tabular form in previous chapter (Table 3) are linked to multiple patterns of the model and consequently share some common remedy actions and tools. In following sections, for clarity and flow of discussion we categorized the findings of our thesis in five sub groups, where we analyze and discuss them with respect to the theory.

5.1 Motivation and Encouragement

One of the biggest challenge for any process improvement activity is to address how employees embrace the change, despite of their natural resistance to change, both at organizational and at individual behavioral level. The clarity of purpose in the process improvement approach is critical for employee motivation and involvement (Middleton, 2001). An improvement program at an organizational level can be a multi-year effort and hence it is of utmost importance to keep the employees self-energized by having tangible benefits to their efforts (Mento et al., 2002). In the case study, the employees were encouraged and empowered to question the status quo and makes small retractable experiments that are believed to refine or streamline the existing processes. These were termed as quick wins and based on the understanding that these experiments might fail but the Lean thinking emphasizes at failing fast so that minimum time is lost and quick reversion to legacy course is possible. At the same time it is equally important to share the lessons learned from these quick wins with other teams and to reward the "quick winners" to promote the Lean culture. In the case study, initial small efforts (quick wins) such as automation for a small set of test cases and developing network-monitoring tool etc. paved the way for bigger improvement tasks while boasting employees confidence and motivation along the way.

Authors find the similarity of this approach from the research work conducted by Kotter (1996) and Mento et al., (2002) as they also include “Create Short term Wins” in their respective step based change models as a critical step for motivating employees for a long-term change effort. In Lean’s terminology, these events are referred as Kaizen events and are marked by achieving small and focused process improvement in an accelerated manner. These events are
characterized by their low investment in terms of cost, time and resources (Toni et al., 2008). Kaizen is a Japanese word for “continuous improvement” and in 1950s, Toyota was able to implement tens of thousands of small improvements each year and the accumulated result of all these small incremental improvements can be seen in form of better quality products and higher customer’s satisfaction (Hibbs et al., 2009).

In the case company, a rewards and recognition program by the name of “High Five” for employees accomplishments with regards to quick wins was established. As well as to promote successful quick wins, specific television screens were mounted in coffee area which projected the quick wins success stories as well as encouraged Lean thinking by displaying various examples.

Process improvement researchers also emphasizes on celebrating the small victories as organizations moves towards larger and drastic changes (Mento et al., 2002). Mento et al., (2002) in their 12-step change model also include “constantly and strategically communicate the change”, as one of the step to stress the importance of communication for engaging employees during change.

It was observed in the case study that repetitive manual testing tasks loomed boredom among knowledge workers and that resulted in increased number of human errors. Practically, automating, (which is one of the pattern in the thesis framework model) the repetitive test cases solve this problem. However broadly discussing, the patterns such as engaging everyone and learn constantly etc. and their respective tools as discussed in Appendix B provides the direction to effectively tackle boredom problem.

Boredom caused by repetitive work has been discussed in literature as an important de-motivating factor for employees (Davies, 1926) (Smith, 1981). A nationwide survey of knowledge workers in South Africa from the financial, science & technology sectors reveal that knowledge workers are looking for continued learning and prefer to work on exciting and challenging projects and easily leave an organization if boredom sets in (Robbins et al, 2003). It is crucial for organizations to continuously invest not only on cutting edge technologies but also on human assets by enhancing their knowledge level to motivate and retain them.

The fundamental characteristic of the all the patterns in Lean-BPR adoption pattern model is to empower the employees by moving the decision making powers to the employees. The results of using this approach in case study where employees are encouraged to make their own decisions regarding quick wins, automation etc. are encouraging.

Researchers like Flinchbaugh (2005) also relates engagement and empowerment of people with decision-making happening at the lowest possible level or at the point of said activity. Deep understanding of the current process is critical before initiating any improvement activity and the people who have to deal with them on daily basis can effectively provide that. Though this fundamental Lean principle is sometimes the hardest for people in managerial positions to adopt, as it requires them to give up power and control. At the same time, the said approach can go wrong if the people start making decisions of their own without having generic rules, working boundaries and guidance to understand the overall effect of their decisions over the whole value stream (Mary and Tom Poppendeick, 2003).
5.2 Coordination and Collaboration

Software development like any other engineering or manufacturing domain is characterized by interaction and collaboration of various teams such as business, marketing, design, development, testing and field support teams etc. The value delivered to the end customer in form of a software product is dependent on how well these teams collaborated with each other to make this whole value stream efficient. The quality of the product in the eyes of customer is measured by analyzing how close the end product is with respect to the customer’s requirements and as well as on the number of short comings or defects. In the case study, the late detection of defects along with frequent changes in customer requirements are some of the main causes for delays and increased costs for the company. As summarized in Table 3, these issues are mapped to deliver fast and learn constantly patterns of the Lean-BPR adoption pattern model. The suggested tools such as working in short iterations, moving towards Pull systems and involving customers in the feedback loop helped in minimizing the said issues effects as shown in case study results (section 4.5).

Over the years, software researchers statistically proved that the misunderstanding of customer requirements as well as defects found further along in the workflow are likely to cost more in order of magnitude than if found in early stages (Tierney, 1993) (Parmar and Shah, 2010). As discussed in earlier chapters, software organizations are moving fast to adopt lightweight development approaches such as Agile etc. The main characteristics of such approaches are their shorter development iteration cycle and shorter feedback loop (Hibbs et al., 2009). Both of these strategies complement each other. Each continuous iteration will result in a working portion of the final product that is fully working, tested and integrated with the earlier iteration part. There will be more valued communication and feedback opportunities among customers, developers, tester and other stakeholders. Feedback comments can be looped back rapidly within the same iteration or the coming iterations to minimize the effort spending on non-value added features along with early fixation of defects. Feedback used in this case study serves two purposes as stated by Van Hilst and Fernandez (2011), i.e. control and learning. It helps in controlling the current process flow on track for meeting the desired objectives as well as it facilitates learning opportunities for continuous improvement of processes based on current experiences.

In this context, Mary and Tom Poppendieck (2003) have also discussed “Pull systems” for increasing collaboration among teams. There are two approaches towards work i.e. either to micromanage the workers by providing them detailed directions (Push system) or to setup an environment where they can figure out themselves what is required of them. In fast-paced environments like software development, it is a serious waste in terms of sending information up in the chain of commands and waits for the directives from upstream. The best way is to “let customers' needs pull the work rather than have a schedule push the work.”. That requires a system that have appropriate signaling mechanism present for the team members to coordinate among themselves and keep the flow of work moving with the next in line customer dictating or pulling the work down. Example of such systems includes Agile tools of Scrum and Kanban.
During analysis of case study company’s current processes, it was observed that test teams were engaged in projects very late. The upstream teams such as design and development teams etc. collaborated with each other and were involved quite early in the projects. They had also started working on small iterations but they did not communicate or share their approach with the downstream test teams. This increased the gap among these teams not only in terms of technical knowledge but also in their team bonding. The model patterns such as engage everyone, optimize the whole, learn constantly and keep getting better and their respective tools such as value stream mapping, feedback, short iterations and frequent lesson learned session helped in bridging the gap among teams and streamlined the workflow. Lean thinking advocates the approach where each team consider their downstream teams as customers and work together to create value as the work flows in their direction (Poppendiecks, 2003). Maglyas et al., (2012) based on their research, presents this scenario from another angle by analyzing work flow process for various medium and large size software organizations. They showed that often in such organizations, end customers are completely ignored after the initial requirement phase and are not engaged until the later stages of the project, which results in project delays and cost overruns. The practical approach as discussed by Hibbs et al., (2009) and also incorporated in the said case study is to hold frequent lesson learned sessions after each work iteration, involving end customers as well. Hibbs et al., (2009) also emphasize on maintaining a lesson learned knowledge base and making sure it is always accessible to all coworkers. This will save the time and effort for all the coworkers as they learn from other’s experiences rather than making their own mistakes. Another suggestion for increased collaboration among teams is to make arrangements for a co-located Integrated Product Teams (IPTs) comprising of team members including customer and stakeholder’s representatives to quickly address the concerns without wasting time and resources (Hibbs et al., 2009). However, this approach was not practiced in the said case study as various teams were geographically separated with each other.

Along with the discussed finding above, issues like over production, extra management tasks and over burden resources are also relevant to discuss in this section as they are also dealt with the same approach of having increased collaboration among all the teams in the case study as directed by the Lean-BPR adoption pattern model. As discussed in theory chapter, like manufacturing, inventories in form of partially done work and/or extra work in software development, which is not needed at that particular time, is a source of waste and over burdening of resources. Lean software development approach demands iterative development where decomposed features are to be designed, implemented, integrated, tested and deployed as rapidly as possible to keep the flow going (Peyton, 2009). Similarly, management activities such as authorization systems, sophisticated project tracking and control systems and release management systems etc. do not directly add value to customers but can be a bigger source of waste in terms of cost, resources and time. These activities can lead to several other wastes like delay, unnecessary handoffs and extra processing etc. in a system. The tools used in this case study such as value stream mapping etc. are recommended by Lean researchers for also identifying wastes in software management processes (Mary and Tom Poppendeick, 2003).
5.3 Variability

The issues found in the said case study such as task switching, waiting in form of handoffs, varying work cycles and frequent changes in requirements are classified under variability in different dimensions and are sources of waste as presented in section 2.1. Appropriate Lean-BPR adoption pattern model tools such as value stream mapping, leveling and decomposition of work in short iteration, establishing pull systems, improving feedback among teams etc. are used to minimize the effect of these wastes.

Software researchers broadly associate the delay problems due to task switching, handoffs and variable work cycles with lack of communication among team members and recommends to work in short iterations with increased feedback opportunities to improve continuously (Hibbs et al., 2009). A software development team comprising of architects, developers and testers etc., not only need clarification among themselves but also from customers and other stakeholders. Let us consider a situation in which a developer comes up with certain questions pertaining to customer requirements and his questions are not answered immediately. This lead to several waste scenarios. The developer can move on to other tasks suspending the one on which he/she was working, making a room for a task switching waste. If the developer start guessing, what needs to be done, potentially a lot of rework will be needed in case of a wrong guess and it gets even worse if the defect due to wrong guess remains in the system for long. Similarly, if the developer waits too long for the answers, the productivity falls to larger extent. Therefore, the solution to such type of wastes lies in increased collaboration among team members.

Irregular task switching on the other hand can potentially be due to mismanagement of both human and capital assets including shared equipment and tools. Goldratt (1997) explained the consequences of this task switching waste phenomenon with a simple example of having same resources for two projects. If each project is expected to take 2 weeks for completion and the second project is started after first one is finished, in total it should take 4 weeks for the said two projects to complete. The problem arises if both projects are started together that require same resources to switch between these projects. Firstly, none of these projects will be completed in 2 weeks and by adding the switching time in between it could easily take up to 5 weeks of work.

The authors of this thesis also observed that frequent and irregular changes in requirements are adding several types of process wastes ranging from task switching, defects, delays and handoffs etc. These are also taken care by moving towards Agile way of working with short iterations and incorporating customers in the feedback loop. Customer needs or requirements are often changed due to number of reasons e.g. customer himself lacks understanding of the system capabilities and limitations, fear of competitor’s products with extra features and also the requirements can be changed due to late detection of defects (Wallace et al., 2004). Ever changing requirements are always a big worry for software designers, developers and testers. They have to switch backward to develop a modified or totally new software code and create new test cases for the change. This uncertainty increases the stress level of employees by over burdening themselves (Nurmuliani et al., 2004). The solution to this issue in eyes of several researchers lies in incorporating efficient change management processes that sees these changes as opportunity to improve the products and processes and not otherwise.
(Pandey et al., 2010). Lean software development patterns ensure close cooperation of customers in the software development lifecycle. As each work iteration results in a working piece of software for customers, they are in better position to judge their own expectations and systems limitations and asks for pertinent and relevant changes. At the same time software resources need to embrace changes in requirements positively and on priority basis with the help of efficient and/or IT assisted change process (Cusumano, 2010).

5.4 Systems Thinking

Lack of overall picture for employees is a prime concern for Lean researchers as it address employees’ inability to understand what is needed from them. At times employees are ignorant of what customers really care about and what they will value. Employees are often too busy in just doing their assigned routine work that they never bother to think at the higher level i.e. to actually questions themselves and assess their contribution towards achieving organizational level goals (Cockburn and Highsmith, 2001). In software industry, finding what value the software bring is not an easy task as software does not have any value of its own, rather the overall value delivered to customers is in the context of larger systems such as an automated car or an online auction website etc and also through the company adopted processes. Another way of looking at the same phenomena is that the value of a software cannot be determined entirely from the development phase, but other phases such as requirements gathering, testing and deployment also play a pivotal role in the overall value stream of developed software (Poppendieck and Cusumano, 2012). Maglyas et al., (2003) in their research observed that often product managers who are responsible for product from conception to realization face difficulties in describing the overall product life cycle. This is because various departments at times work in an isolation and independently on their own chunks of work without focusing on the full product and its value in eyes of end customers.

In the said case study, the suboptimization problem is visible when the development teams adopted the Agile way with shorter iterations and working on specific features for particular iteration, whereas downstream teams were not kept in the loop for this improvement initiative. This seems to be negating the Lean-BPR adoption model patterns of optimize the whole and engaging everyone, as some of teams move towards optimizing their part of work flow and overlooking the whole software development lifecycle flow for faster delivery.

Van Hilst and Fernandez (2011) explain systems thinking as “a shift from traditional mechanistic, cause-effect thinking to a holistic view of interrelated contributing factors”. Instead of narrowing down to a single cause, it broadens the canvas by analyzing more factors and their relationships. Earlier organizations, likewise mechanical systems said to be modeled for a controlled environment that assume processes are predictable. However development especially software development is not always predictable and demands an open environment. Researchers like Bertalanffy and Deming who are considered founders of systems theory have background in agriculture and biology; have compared and applied systems theory concepts together with that of living systems (Van Hilst and Fernandez, 2011). For example, no one can tell a crop how to grow or cows how to milk but studying and analyzing the factors that contribute to yield and productivity can be improved. In organizations, where problems and opportunities are dynamic
Most importantly by nature, software is also not limited to a single time-bound effort; modification and maintainability of code base over time are important factors too. Hu (1999) also compared sustainable software systems to characteristics of living organisms that can develop, maintain, reproduce or at times renew itself by effectively utilizing the material and energy from the surrounding environment. The patterns of Lean-BPR adoption pattern model namely optimize the whole, engage everyone and keep getting better as presented in this thesis also envision the employees as part of living system organizations where they are empowered to act individually or in groups and have the capability to alter the process to achieve their improvement goals.

Another aspect, which the case study reveals, is the role of communication strategy for an organization. The employees were aware of general Lean principles via various trainings but they were not knowledgeable when it comes to application of Lean philosophy in software development. This gap can be filled by proper communication and strategy from the top leadership. Although Lean is known to be a bottom-up approach but commitment and support from top leadership is equally important in promoting and encouraging Lean thinking across the organization (Mary and Tom Poppendeick, 2003). Coaching, mentoring and trainings for employees for empowerment are the way forward for a self-learning living organization (Van Hilst and Fernandez, 2011).

5.5 Automation
Excessive paperwork consumes resources, slows down the response time when it requires meaningless sign-off and approvals and at times these documents are not even read or consulted fully by anyone including the customers. This eventually results in adding little or no customer value. In the case study, the solution to these issues are looked through the patterns build quality in and BPR along with their respective implementation details. The important dimension of these patterns is to evaluate how to embed IT in to software development life cycle to create value for the customers (Charette, 2003). That is where BPR concepts came in to help as IT tools can be of great assistance to analyze, restructure and automate several manual processes including test, integration, build, configuration management and release processes. They help in making decisions more quickly and resolve problems as soon as they start to appear (Hindle, 2008). The 4-step BPR process used in this case study helped in identifying various processes, which needs automation such as testing, requirement management and test case management etc. However, the fundamental point to note here is that all the BPR related steps are initiated by the empowered employees themselves in a bottom-up approach and in combination with other patterns of Lean-BPR adoption pattern model.

Researchers like Limerick and Cunnington (1995) argued quite early after BPR’s introduction that the strengths and success of BPR depends on the empowerment of the individual; while most of the earlier BPR literature of 90s is focused on IT and automation while neglecting the complex human, cultural and organizational elements. Human assets with specialized education
and training (e.g. developers, testers, support technical staff etc.) combined with the technical equipment and tools are the critical elements in the efficient running of the software development process. Wickramasinghe et al., (2013) termed it as a socio-technical approach that motivates the people to use their own knowledge, wisdom and their desire to make effective use of the technology in quest for improving the legacy processes.

In connection to test coverage, manual repetitive testing effort contributes to increased cost, time and resources throughout software development lifecycle. It is estimated that depending on software product criticality and complexity, testing itself can bear in the range of 30-60 percent of all product life cycle costs (Polo et al., 2013). Typically that costs sums up all the effort used to detection and fixation of defects. While discussing early defect detection, defect prevention cannot be ignored either. Build quality in and BPR patterns in the Lean-BPR adoption pattern model with help of their respective tools addresses defect prevention approach. One of the Lean tool used successfully by Toyota earlier is called Poka-yoke, a Japanese term meaning "mistake-proofing" (Shingo, 1981). It is based on the principle that “defects occur when the mistakes are allowed to reach the customer”. The main idea is to identify and eliminate errors as close to the source as possible. In the case study automating, the regression tests in conjunction with partial software delivery in each iteration ensure not only quick discovery of errors but also preventing many defects that can occur later as a byproduct of these detected errors.

However, several case studies on software organizations shows that test automation raises one’s hopes to an higher extent yet often they fail to deliver the real benefit in terms of cost and utilization of resources (Pettichord, 2001). Pettichord (2001), while presenting his seven steps of successful test automation pointed out lack of clear goals, lack of experience and too much focus on technology as major causes of automation failures. Both practitioners and researchers agrees that not all test cases can or should be automated, and careful consideration is needed while selecting test cases for automation (Dustin et al, 1999). The 4-step BPR process as pattern of Lean-BPR adoption pattern model used in the case study for implementing test automation ensure proper analysis of test process, identification of areas where automation is necessary and can add value to overall software development process and lastly but importantly, constantly assessing the results for continuous improvement.
6 Conclusions and Implications

This chapter accounts for the thesis conclusions based on the data and analysis performed and relate them to the existing research. It also discusses their implications along with the research limitations and suggestions for potential future research on the said subject.

The main purpose of this thesis study was to evaluate if and how Lean thinking is applicable in software development industry and how BPR can be implemented under a bottom-up Lean framework for SPI. The authors of this Master thesis have investigated and analyzed the impact of adopting Lean software development and BPR principles for software process improvement in a case study. Inspired by Van Hilst and Fernandez’s (2010, 2011) systems of patterns, authors derived the Lean-BPR adoption pattern model with seven established Lean software development principles and a 4-step BPR process, as its patterns. The associated tools and activities (Appendix B) of these patterns are then applied on the case study company to identify and resolve the major challenges in the overall software development lifecycle. The results achieved in terms of improved KPIs after initiatives based on Lean-BPR adoption pattern model are presented in section 4.5.

6.1 Conclusions

The review of the relevant literature as well as the findings from the case study support the notion that Lean principles can be effectively used for software process improvement and can also complement BPR approach mainly through employee empowerment and their involvement in the entire process.

It is also worth mentioning that the eight patterns of Lean-BPR adoption pattern model are not mutually exclusive. These patterns have different strengths in different types of situations. In addition, they exhibit different strengths even in the same scenario. They may share similar tools for dealing a particular challenge in improving software processes. Van Hilst and Fernandez (2011) also highlighted the interdependencies among different patterns of their system of patterns. They also concluded that the best examples of process improvements are those where these patterns are applied in combination. On the other hand, their research showed negative impact on processes where one of the patterns is used and others are excluded.

In this thesis, an attempt has been made to incorporate IT centric top-down BPR approach with the bottom-up Lean framework. Contrary to traditional BPR approach, where there are specific BPR leadership teams; 4-steps BPR initiatives were taken by the employees themselves for use of IT and automation. It is also observed that the 4-step BPR pattern is quite similar to the steps of Lean value stream mapping tool. This is consistent with the findings of Ricondo and Viles (2005), who investigated the working relationship of Lean and BPR approaches. They conclude that both Lean and BPR have some basic similarities in their management and quality tools and they can complement each other if practiced together. Childe et al., (1994) also support the bottom-up restructuring of processes and prefer incremental changes under Lean framework rather than radical changes to improve processes.
The SPI initiatives based on Lean-BPR adoption pattern model clearly indicate that *suboptimization* (optimizing only some processes and not the overall end-to-end process) is unavoidable in the initial stages of improvement process. However, as practiced in the case study, these improvement initiatives should be gradually extended to all the other segments of the entire value stream of software development lifecycle. Research literature also consider suboptimization as a dangerous activity if certain processes are improved and the others are left untouched or assessed (Mary and Tom Poppendeick, 2003). According to Van Hilst and Fernandez (2011), in the final analysis, any small improvement should be treated as an important step as all these multiple improvements add up to achieve the bigger goal of strategical level process improvements.

In the case study, Lean initiatives such as quick wins etc. are carried out to motivate and empower employees to analyze and improve their own working processes. The knowledge workers have been empowered to make their own decisions such as selection of IT tools etc. Similar strategy can be seen in Middleton’s (2001) case study where he experimented to move the responsibility for measuring quality from managers to the workers. It resulted in a much quicker and through response to defects. Lean researchers agrees that the success of Lean lies in the changed organizational culture where employees are given a sense of ownership of the tasks they do along with the confidence that employees themselves are the best resource to improve their working processes (Walley et al., 2006).

It is also observed that the testing department often became the bottleneck due to ever-growing testing queues and limited resources. This issue is tackled with relevant tools and methods of thesis framework model such as shorter iterations where only the relevant features are tested as soon as they are coded and with increased feedback sessions etc. Middleton (2001) carried out a similar research on two case studies of improving software processes by applying Lean software development principles. In his conclusion, Middleton (2001) also make the software test teams responsible for software quality problems and urge them for obtaining fast results from low cost actions. Middleton (2001) uses the same approach with positive results as authors in this thesis used for reducing the inventory of requirements and test cases while working on shorter work cycle for quick discovery of faults.

### 6.2 Implications and recommendations

While the main focus of this thesis was to evaluate if Lean software development and BPR principles are applicable for SPI, the analysis was founded on academic theory. Most of the existing literature focuses on the practical tools and techniques used for application of Lean software development principles and BPR for SPI. The main theoretical contribution of this thesis is the derivation of a Lean-BPR adoption pattern model from established system of patterns for process improvement. This theoretical framework was used to identify and classify the issues encounter during process improvement initiative in the case study. In order to perform a theoretically supported analysis, these issues are linked to the patterns of the Lean-BPR adoption pattern model (Table 3) and appropriate tools and actions (see Appendix B) are taken to rectify them and to visualize the practical implications of the framework.
From this study, no inherent reason has been found to suggest that Lean software development principles and BPR cannot be used for software process improvement. The authors see a clear need for education and training of employees with regards to Lean software development principles. There is definitely a trade-off here as these trainings are a cost to the organization. From leadership perspective, it is always tempting to invest funds on buying new IT assisted tools to solve productivity problems. But as shown in this experiment, it is more about the empowered and motivated people and how they embrace Lean principles along with supported IT tools for process improvement.

During the writing of this thesis, authors have identified several directions for further research. The gathered data in form of results indicate some validity to the theoretical framework of Lean-BPR adoption pattern model, yet it is not rich enough to be conclusive. Practically, similar experiments could be carried out by SPI researchers as enough details about the steps are provided to repeat the experiments. Furthermore, as this study was an exploratory exercise, based on the input of limited internal participants of a large software organization, it would be interesting to research whether the thesis conclusions still hold for external software organizations of varying size e.g. small and mid-size software organizations.

Another potential area to research is analyzing the interdependencies of patterns of Lean-BPR adoption pattern model, as this study focuses more on the patterns and their combined effect on the KPIs improvement.
7 References


8 Appendix

8.1 Appendix A

Lean implementation success stories in various sectors

One of the notable industry that has benefited by adopting Lean in recent times is Healthcare sector. Some of the major challenges faced by healthcare sector worldwide today are the escalating costs of providing care, increase in the population & chronic diseases and growth in medical technology solutions (Graban, 2011). They also often suffer from skilled employee shortages, which often resulted in over burdening and stressing of the existing medical human resources, risking the life of patients (Gaba and Howard, 2002). Some of the hospitals tackled these challenges by adopting Lean initiatives in the operations to provide quality services to their patients and to eliminate wastes that commonly exist in their administrative processes (Koning et al., 2006). Some of the examples are listed below where Lean methods have positive impacts on hospitals throughout the world:

- Akron Children’s Hospital in Ohio, USA undergoes a Lean transformation after identifying the problem of long patient wait times for MRI exam. Lean tools such as VSM helped the multidisciplinary team of people from various departments e.g. from laboratory personals, accountants, medical staff, suppliers and patients themselves to identify the root causes and implement proposed changes rapidly. The result was not only reduced waiting times for patients for MRI exam but also $1.2 million increased revenue after the transformation (Chand and Musitano, 2011).

- Similarly, ThedaCare medical center in Wisconsin reduced the patient waiting time for orthopedic surgery from 14 weeks to just 31 hours from the first call to the surgery department (Gabran, 2011).

- The Red Cross Hospital in the Netherlands successfully integrated the Lean with Six Sigma statistical process improvement techniques and completed number of significant Lean Six Sigma projects with minimum savings of €20,000 per project (Koning et al., 2006).

Supply chain business is another example of businesses that has been benefited by adaptation of Lean philosophy. Supply chain is a complex set of organizations, services and information that involved in moving a product or service from a source to the customer. It is comprised of several key components such as suppliers, procurement, manufacturing, warehousing, transportation and the end customers themselves (Tompkins, 2010). Due to the involvement of many parties in the process, it require careful cooperation among these entities to maintain the process flow. Lean principles guide the overall Supply Chain Management (SCM) to increase the speed and responsiveness to the customers, reduce excessive inventories as part of eliminating wastes and reduce the overall cost (Sabri and Shaikh, 2010). Lean SCM organizations are known for not only reducing the finished goods inventory but also reducing the raw material inventories by arranging just in time deliveries of raw material from the preferred suppliers to support manufacturing (Tompkins, 2010).
8.2 Appendix B

- **Optimize the whole**
  - Tool 1: Measurements
  - Tool 2: Contracts

- **Eliminate waste**
  - Tool 3: Seeing Waste
  - Tool 4: Value stream mapping

- **Build Quality in**
  - Tool 5: Perceived integrity
  - Tool 6: Conceptual integrity
  - Tool 7: Refactoring
  - Tool 8: Testing (Automated)

- **Learn constantly**
  - Tool 9: Feedback
  - Tool 10: Iterations
  - Tool 11: Synchronization
  - Tool 12: Set-based development

- **Deliver fast**
  - Tool 13: Pull systems
  - Tool 14: Queuing theory
  - Tool 15: Cost of delay
  - Tool 16: The last responsible moment

- **Engage everyone**
  - Tool 17: Self-determination
  - Tool 18: Motivation
  - Tool 19: Leadership
  - Tool 20: Expertise
  - Tool 21: Making decisions

- **Keep getting better**
  - Tool 22: Options thinking

Figure 25: Association of 22 Tools with 7 Leans Software Development principles (Source: Mary and Tom Poppendeick, 2003)
8.3 Appendix C

Figure 26: Ford Example of Legacy accounts payable process (Source: Hammer, 1990)

Figure 27: Ford example of re-engineered accounts payable process (Source: Hammer, 1990)
8.4 Appendix D

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>Risk taking will be embraced by your management team, not punished.</td>
</tr>
<tr>
<td>Easy to Implement</td>
<td>Making the change or improvement does not require a great deal of coordination and planning.</td>
</tr>
<tr>
<td>Fast to Implement</td>
<td>Making the change or improvement does not require a great deal of time.</td>
</tr>
<tr>
<td>Within the Team's control</td>
<td>The team and its management are able to gain the support of the people needed to make the change. The scope of the change is within the team's ability to influence.</td>
</tr>
<tr>
<td>Easily Reversible</td>
<td>The quick win opportunity should be reversible if the opportunity requires more time and resources than originally thought.</td>
</tr>
</tbody>
</table>

*Figure 28: Criteria for defining "Quick Win" at Motorola Solutions (Source: Lean Six Sigma Green Belt Training, 2008)*