2020-05-04

Designing Software Process Improvement Framework for Small Scale Software Firms

Mengiste, Tadele

BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

SCHOOL OF RESEARCH AND POSTGRADUATE STUDIES

FACULTY OF COMPUTING

Designing Software Process Improvement Framework for Small Scale Software Firms

Tadele Mengiste Eshete

BAHIR DAR, ETHIOPIA

May 4, 2020

Designing Software Process Improvement Framework for Small Scale Software Firms

Tadele Mengiste Eshete

A thesis submitted to the school of Research and Graduate Studies of Bahir Dar Institute of Technology, BDU in partial fulfillment of the requirements for the degree of

Master of Science in Software Engineering in Computing Faculty.

Advisor: Gebeyehu Belay (Dr of Eng.)

Bahir Dar, Ethiopia

May 4, 2020

DECLARATION

I, the undersigned, declare that the thesis comprises my own work. In compliance with internationally accepted practices, I have acknowledged and refereed all materials used in this work. I understand that non-adherence to the principles of academic honesty and integrity, misrepresentation/ fabrication of any idea/data/fact/source will constitute sufficient ground for disciplinary action by the University and can also evoke penal action from the sources which have not been properly cited or acknowledged.

Name of the student Tadele Mengiste Signature

Date of submission: 22/08/2012 E.C

Place: Bahir Dar

This thesis has been submitted for examination with my approval as a university advisor.

Advisor Name: Gebeyehu B (D.)

Advisor's Signature:

Bahir Dar University

Bahir Dar Institute of Technology-

School of Research and Graduate Studies

Faculty of Computing

THESIS APPROVAL SHEET

Student:

Tadele Mengiste — 22/08/2012 e.c

Name — Signature — Date

The following graduate faculty members certify that this student has successfully presented the necessary written final thesis and oral presentation for partial fulfillment of the thesis requirements for the Degree of Master of Science in Software Engineering.

Approved By:

Advisor: Gebeyehu B (D.I) — Signature — Date

Name

External Examiner: Million M. — Signature — Date

Name

Internal Examiner: Mekonnen W (Ph.D.) — Signature — Date

Name

Chair Holder: Gebeyehu B (D.I) — Signature — Date

Name

Faculty Dean: Belete B. — Signature — Date

Name

To my father and mother

ACKNOWLEDGMENT

First and foremost, I would like to express my deep and sincere gratitude to my Advisor Dr. Gebeyehu Belay, who provided me with this opportunity to work under his supervision. His guidance, encouragement, motivation and constructive criticism helped me to learn and improve a lot throughout this thesis.

I would further like to thank Dr. Million Meshesha and Dr. Mekonnen Wagaw for their participation in my MSc thesis defense and for their kind and motivating comments to enrich my research work.

I would like to thank everyone involved in the pre-research survey at Ethiopian Airlines Aviation Academy software developers' team (developers as well as the management) for their thorough and tireless guidance, in particular Wondwossen Gebreab and Dawit Alemayehu for answering all my questions and taking the time to help me, despite their busy schedule. This saved me a lot of time and effort and improved the quality of the thesis.

I would thank those three software team leaders and their project teams as well whom I observed and interviewed at Bahir Dar ICT Business Incubation Center including the customers of these firms.

I would also like to thank my friends working in Bahir Dar University, for their support and guidance throughout this master's degree. Finally, my loving thanks are due to my families for their encouragement, support and prayers throughout my studies.

ABSTRACT

This study proposed a simplified software process improvement framework for small scale software firms. It identifies a set of practices core to software development. It can be more readily adopted by small scale software firms, particularly by those in developing countries settings with limited resources where the context and constraints may differ from more developed countries. These programs often do not state how to approach the process improvement program, too cumbersome and costly to implement by considering Ethiopian software development culture. To address these issues, a new framework developed integrating the features of Capability Maturity Model Integration version 2.0 with DevOps to clarify how software practices inside firms can be improved which should be more accessible and less costly to implement. The combination of CMMI and DevOps enable firms to take advantage of models' strengths and compensate for their weaknesses in enhancing continuous software process improvement and rapidly responding the customer's changing needs by covering as much key process areas as possible. The framework helps those firms by increasing the quality of the software developed to their customers. According to the results obtained both the firms A and B improved from managed to defined level, and firm C is at defined level which doesn't achieved a significant level of improvement after the implementation of the developed framework. The overall maturity of key process areas of CMMI V2.0 categories by percentage before and after the intervention for firms is 63.45 and 74.31 for firm A, 61.31 and 73.59 for firm B, and 66.13 and 73.24 for firm C respectively. In addition to the results of CMMI V2.0, the firms were also assessed from DevOps perspective and achieved improvements by percentage before and after the intervention for firms as 66.5 and 77 for firm A, 72.33 and 83.33 for firm B, and 71.5 and 81 for firm C respectively. According to the results obtained firm B and C improved from defined to quantitatively managed level while firm A records a significant level improvement from managed to a defined level. In general, the framework results in a significant improvement of the firms' maturity of the software development process for firm A, 10.86% and 12%; firm B,12.3% and 11%; and firm C,7.11% and 9.5% improvements for CMMI V2.0 and DevOps maturity respectively. Generally, the framework results in a significant improvement of the firms' maturity of the software development process.

Key Words: *Software Process Improvement, CMMI V2.0, Capability Maturity Model Integration, DevOps, Framework, DSRM, Design Science Research Methodology*

TABLE OF CONTENTS

LIST OF FIGURXXES

LIST OF TABLES

LIST OF ABBREVATIONS

CMMI	Capacity Maturity Model Integration Version 2.0
CSAT	Customer Satisfaction
DevOps	Development-Operations
FIRM A	Boost software development plc
FIRM B	Abebe and his friends' software development S.C
FIRM C	Ethiopian Airlines software development team
IDI	ICT Development Index
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
KPAs	Key Process Areas
SEI	Software Engineering Institute
SME	Small and Medium Enterprises
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination

CHAPTER ONE

INTRODUCTION

1.1 Background

Since the last decades, software products have gained an essential role in our daily life. This condition demands to be of optimal in terms of quality. To produce high quality software many practitioners and researchers put more attention on the software development process. IEEE Computer Dictionary (1991) defines process as "a sequence of steps performed for a given purpose". According to Paulk et al. (1995), a software process can be defined as "... a set of activities, methods, practices, and transformations…" that developers use in the development and maintenance of software and other associated products such as project plans, design documents and code. Large investments are poured to improve the software development process. A good software development process is the best way to assure quality of software products.

To maintain and increase competitive advantages, software organizations must continually strive to refine and improve their development processes. Sommerville (2011) states that a development process should be updated improved and maintained to meet current business and customer requirements. Zhang and Shao (2011) explains process improvement as simply: "consistently apply the practices that give you good results and change the practices that cause problems". The motivation behind SPI is to achieve business goals and develop quality products through a mature process and practices of process areas. A Process Area is a cluster of related practices in an area that, when implemented collectively, satisfy a set of goals considered important for making significant improvement in that area. CMMI V2.0 Process Areas includes doing, managing, enabling, and improving (Sommerville, 2011). In this thesis, the literature on software process improvement is reviewed. Then the current software development process at three small-scale Ethiopian software development firms is analyzed.

Many universities (SEI, 2010), research centers (Sommerville, 2011), and associations (Turner Consulting Group, 2007) around the world have tried to find their own answers to this issue by proposing software process improvement frameworks that are dedicated for the use inside small-scale software developing firms. However, the proposed solutions were still too complicated and cannot be applied easily by these firms since it requires special skill to adopt and implement (Niazi, & Babar, 2009; Pino et al., 2010). None of them represented a solution that fits small-scale firms; moreover, all of these frameworks were built such that they fit the country from which the

information was gathered. Even if there doesn't exist a unique definition of a small-scale software company, in Ethiopia a frequently used definition is that a firm employing less than 20 people (ABS, 2002). Small-scale enterprises are those business enterprises with a paid-up capital below 1.5 Million Birr. Small-Scale businesses are driving force for economic growth, job creation, and poverty reduction in developing countries (Arega et al., 2016). Small-scale software firms play a fundamental role in most countries' economies, and they represent up to 85% of all software firms in the US, Canada, China, India, Finland, Ireland, and many other countries (Richardson, 2002).A new SPI framework integrating CMM V2.0 and DevOps Methods is developed by combining the best features of previous approaches and addressing their limitations by considering small scale software developing firms. This study offered a simplified SPI framework that identifies a set of practices core to software development and which can be more readily adopted by small scale software developing firms, particularly by those in developing countries settings where the context and constraints may differ from more developed countries. Once these core SPI practices are institutionalized, organizations can then extend their SPI programs to embrace other industry standards.

The CMMI framework consists of five levels of software maturity (Sommerville, 2011). The first one is initial, in which the process is ad hoc, informal, not implemented, and unpredictable. The second level is managed which is implemented only in a single team, works reasonably well in a project, and often reactive. The third maturity level of CMMI is defined which is standardized and implemented within the whole organization. Quantitatively managed the fourth level, have a widely deployed, repeatable, measured, stable, and benchmarked characteristics. The highest (fifth) level optimized known by its properties like continuously improved, root cause analysis, and resolution. According to Lwakatare (2017), DevOps maturity can be measured in four distinct areas: culture, automation, measurement, and sharing. A formal assessment of the current software development process considering CMMI V2.0 categories and DevOps dimensions was gathered based on the questionnaire designed and semi-structured interview by selecting promising small software firms. Some improvement suggestions on the key software practice areas have been made based on the assessment values calculated and the effect of the best practices suggested by the framework have also assessed after the intervention.

1.2 Statement of the Problem

Software Process Improvement (SPI) has become the key to the survival of a lot of software development firms. However, most of its benefits tend to accrue more easily to large organizations that have the resources to implement SPI programs (Turner Consulting Group, 2007). However, small software firms, specifically those in developing countries, are yet to adopt and benefit from these initiatives (Kunda, & Brooks, 2000). One reason why SPI initiatives are not widely adopted and used by small firms is the lack of awareness of such programs and/or the benefits of these programs and interventions (Chevers, & Duggan, 2010). Second, the assessment of maturity level and improvement plans of SPI programs are often too awkward, time consuming, disruptive, and costly for small organizations to implement (Niazi, & Babar, 2009; Pino et al., 2010). Small firms often face budget, time, and resource constraints when they approach process improvement. The SPI framework is developed for small software firms as these firms do not have the resources and cannot bear the cost to implement the standards that are designed to be used by large firms such as CMMI, SPICE, and ISO. (McCaffery et al., 2007). Brodman and Johnson (1998) reported that the cost of CMM implementation was prohibitive for small firms. It is usually also considered not only costly but also hard to define, formalize, enact and institutionalize the frameworks since formalization requires specializing skills about software processes, mastering special notations and tools.

Complexity, adapting issues, difficulty to implement and interpret, and require lots of organizational changes are also included under the reasons that prohibit small scale software firms from adopting SPI initiative (Dangle et al., 2005, Khan et al 2010, Lukasiewicz and Miler, 2012, Zhang and Shao, 2011). Small software firms have different characteristics compared to other sizes of software firms such as number of employees, capital, recourses, tools used, methods adopted. (Hofer, 2002). Cater-Steel (2002) summed up the characteristics of small software firms compared with large software organizations from the verity that these firms have a flat structure, less formalized decision-making structures and procedures, and provide more freedom for employees to depart from the rules. In these firms; just one or two person has\have made serious managerial decisions. These firms neglect the training compared with larger software firms. Moreover, although senior management or middle managers in these firms realize the benefits of process improvement and are willing to devote effort, they lack guidelines of effective approach for the improvement project. To sum up, there are no simplified SPI models tailored to suit the norms and constraints of small software development firms. Small-scale firms need external assistance to adopt and implement standards (Laporte et al., 2008a). This represents a potential gap, as prior studies suggest that for process improvement models to be useful, they need to match the culture of small-scale software development (i.e., with short

deadlines, dynamic projects, and tight budgets) and their settings (Oktaba et al., 2007; Scott et al., 2001).

It is noteworthy that Ethiopia is a large and diverse nation with many linguistic, cultural, and genetic diversity. Cultural diversity mainly considered as an issue from the customer perspective rather than from the vendor or supplier perspective (Deshpande et.al., 2010). The study provides an account of the investigations from the Ethiopian small software firms perspective, envisaging how the nation's cultural, religious and linguistic diversity reflects on the establishment and operation of Ethiopian based small software development firms. According to Degif et.al, (2016) the software development industry in Ethiopia is in its early stage. The software development scenario in Ethiopia is like a rat race where nearly every young person you see in the Tech space in Ethiopia is a mobile app, web app developer and a startup founder. This is certainly a saluted development but what it's seen on ground right now goes beyond that. Several young developers have really good applications that are set to solve real issues in Ethiopia like location based apps in tourism sector, traffic focused apps, eduApps, mHealth apps but they are always stuck after development of the app. How to move to the next level of making the app known and then becoming a profitable venture by deploying the app into operation is a hard nut yet to be fractured. Full adaptation of the technology has been difficult in Ethiopia due to a restrictive information seeking culture, and diverse local languages, scripts and dialects which makes software development difficult to build user friendly interfaces in the local languages for online connectivity for promoting information sharing and developing a successful product nationwide in Ethiopia (EITPA, 2020).

According to Niazi et al (2003) much attention has been paid to "what activities to implement" instead of "how to implement" these activities. It's believed that identification of only "what" activities to implement is not sufficient and that knowledge of "how" to implement is also required for successful implementation of SPI program. For these reasons, DevOps principles and practices were integrated with CMMI that gives an explicit guideline for how to develop software product at a project level. DevOps is concerned with addressing this challenge through better collaboration between development and operations. Together, these models form a comprehensive framework for structuring the software firms by improving their software processes. Interpreting CMMI for special cases like small firms, addressing as much KPAs as possible, and achieving specific business goals is a challenging job. Identifying the most important process areas and explaining their specific goals and specific practices of CMMI for small-scale software developing firms and developing a new framework for the improvement of processes in small-scale software developing firms is the goal of this research.

Therefore, the research questions that are answered by the thesis are formulated as follows:

- ✓ What is the impact of adopting CMMI V2.0, and DevOps towards process improvement for software quality in small scale firms?
- ✓ How to define the extent of software process improvement?
- ✓ How to develop a framework for software process improvement using DevOps and CMMI V2.0 for small-scale software developing firms for getting quality software?
- ✓ How to differentiate DevOps and CMMI V2.0 for software process improvement towards small-scale software firms?

1.3 Objective of the Study

The general and specific objectives of the research are discussed in this section:

1.3.1 General Objective

The main objective of this study is to develop a software process improvement framework for small scale software firms using CMMI V2.0 and DevOps.

1.3.2 Specific Objectives

To accomplish the above stated general objective, the following specific objectives are formulated.

- ✓ To explore small-scale software developing firms' views and opinions about their current CMMI V2.0 and DevOps practices by conducting interviews and use of questionnaire.
- ✓ To enable small-scale software developing firms to assess their current software process for improvement.
- ✓ To define the way CMMI V2.0 and DevOps would be combined to improve small-scale local software developing firms' process.
- ✓ To propose a process improvement framework that is dedicated for use inside small-scale software developing firms.
- ✓ To identify the relationships between software development approaches to software process improvement best practices.

1.4 Scope and limitation of the Study

The focus of this study is on developing a software process improvement framework for small-scale Ethiopian software developing firms based on CMMI V2.0 and DevOps. Formal assessment has carried out to assess the maturity level of firms. After performing formal assessment if the firms need some process improvement, the framework were applied practically. The effect of the framework on these small-scale firms is measured by static and dynamic validation like expert opinion and pre-post comparison methods of evaluation. The study covered all the practices occurred under CMMI V2.0 and DevOps categories. In order to complete the thesis within the allocated time frame, some limitations were faced. Since there is no generally accepted framework exists for DevOps adoption and implementation, integration with CMMI is the challenge (Lwakatare, 2017). The study only covers the CMMI and DevOps Maturity level of each firm. It doesn't cover the process metrics like efficiency, productivity, cycle time, turnaround time, throughput, error rate, and cost effectiveness.

1.5 Significance of the study

This thesis is significant for several reasons. The main reason to do this research is to identify and address the problems related to software process improvement in small-scale software developing firms' projects in Ethiopia. The research highlights the major obstacles for the process improvement for small-scale software developing firms and provides solutions based on CMMI V 2.0 and DevOps. The research also identifies the key process areas for small-scale software developing firms and provides process improvement guidance in those relevant process areas of these firms. Interpreting CMMI for small-scale software developing firms helps firms to achieve better results and build quality-oriented products, which saves both time and cost. This also helps small-scale firms to adopt a process improvement framework with practical implementation methods. Since scholars have identified several benefits of SPI programs including improvement in software quality and customer satisfaction, and reductions in project cycle time and development costs (Niazi et al., 2010; Krishnan and Keller, 1999; Staples and Niazi, 2008).

The framework developed also enhances small scale software developing firms to improve some of their processes, developers, because they can take control and pride in their work, and customers, because they get better quality. The main advantages of software process improvement according to Regan (2011) includes software quality improved, productivity increases, low quality cost reduced, satisfaction of customer increases, consistency is improved in budget and schedule delivery, cost of software development reduced, and team spirit improves. This is considered as the first step to

introduce a quality system in these firms. The research also points out the need to identify benefits and challenges of CMMI and DevOps adoption and implementation on small software firms. The process improvement framework and its questionnaires can be used in assessing maturity of software firms towards their process. In addition, it can serve as a guide in designing improvement efforts and implementing process improvement. The maturity assessment results of the thesis can be used as initial benchmark information in prioritizing and designing improvement actions. Firms have used the framework to study their own current operations and identifying the highest priority areas for that needs improvement.

Scientifically, the research contributes to researches about software process improvement especially focusing on small firms for developing countries like Ethiopia. It has a significant contribution in software process improvement by integrating CMMI V2.0 as model for guiding the process improvement and DevOps as a software development method so that other researchers can try integrating CMMI V2.0 with other software development methods like the agile ones or other models for process improvement with DevOps which best suits for their settings.

1.6 Organization of the thesis

The thesis consists of five chapters. The first is introductory chapter that contains descriptions on the area of software process improvement by clearly elaborating on the necessity of the framework under small scale software firms. This chapter also contains the statement of the problem that deals why it's needed to work on this area, the objective (general and specific) that focused on the descriptions on what is to be achieved by the study, the scope and limitation of the study in which the area and the amount of information to be included in the study are defined, the significance of the study that determines what others are benefited and/or how readers advantageous or learn from the study, and finally the organization of the study that clarifies the contents to be included in each chapters of the study. The second chapter contains review of related literature which includes critical analysis of relevant existing knowledge on software process improvement framework for small firms including related works with regard to software process improvement, small scale software firms, and DevOps approach. Chapter three is concerned with the research methodology that presents the choice of the CMMI V2.0 and DevOps approaches, survey population characteristics and data collection approach. It clarifies what was done and how it was done, what data was recorded, the tools or instruments used in data collection and the methods of analyzing the data. The development of the framework that presents the components of the developed SPI framework and how it was designed and its implementation guidelines, results of the framework implementation, analysis &and

evaluation results obtained due to the implementation of the framework are discussed under chapter four. The fifth chapter which is the closing chapter that focuses on concluding the study results through an articulation of the research findings and provides recommendations for the intended reader by providing information as to how the research results are beneficial for future research.

CHAPTER TWO

LITERATURE REVIEW

2.1 Software and Software Process

Nowadays, software has become a highly valuable asset in business and economic environments (Basri, and O'Connor, 2011). According to Xu and Sjaak (2007) a software process can be defined as "a packaged software component configuration or a software-based service that may have auxiliary components and which is released and exchanged in a specific market". Here packaged components refer to all kinds of programs. The software product takes different forms (Lukasiewicz and Miler, 2012): small, COTS (Commercial Off-The-Shelf Components), packed software, large commercial software, open source software and services. It is important to remember that processes only exist in the real world. They can be observed, described and reflected upon, but a process is always equal to a sum of real-world actions. These actions need not be purely physical; observation and reflection are examples of actions with no physical component. When we observe the process, we build an abstraction of the process (see figure 2.1). This abstraction is what we use when we improve the process; however, it is usually not part of the process itself.

Figure 2.1 A process and its abstractions (Sigurd, 1997).

With this distinction in mind, software process is defined as the actions performed by an organization in order to produce software (Sigurd, 1997). According to Sigurd (1997) software processes do not exist in a vacuum. They are always executed by an organization; this is depicted in figure 2.2. The actions of the process are done by people, possibly playing different roles. It produces a set of products, consuming resources during execution. It provides project members a regular method of using the same way to do the same work. To address the issues of software development quality, scholars focused on three factors deemed to be key determinants of quality (Biro and Messnarz, 2009; Niazi, 2012; Gorla and Lin, 2010), which are in need of specific attention to address the crisis. These are: people, technology, and the IS development process. Some have further argued that of

these, the software development process is the most influential and the “glue” that ties the triad together (i.e., people–process–technology) (Humphrey, 1989; Paulk et al., 1995; SEI, 2010).

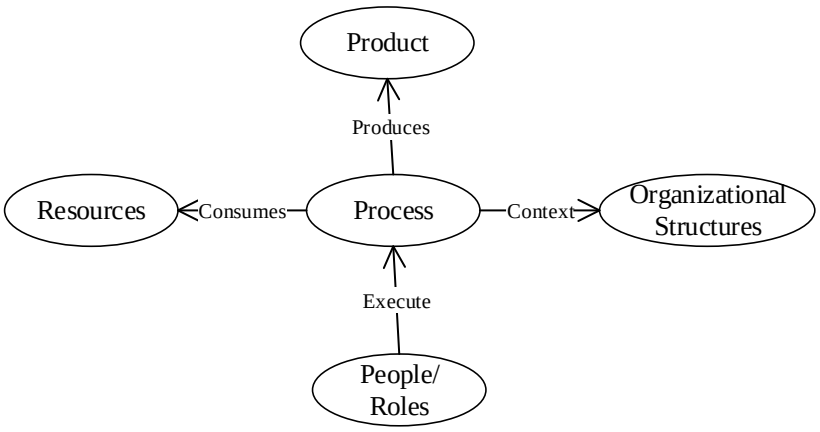


Figure 2.2 A process and its environment (Sigurd, 1997)

In this thesis, a software development process, or software process is referred as a series of actions, procedures or activities performed by the organization in order to design and produce a software product. The formal and more detailed definition of software development process is a process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use (Kuilboer and Ashrafi, 2000).

Software development process, commonly referred also as software process, is a set of activities and guidelines that lead to the production and maintenance of a software product (IEEE Computer Dictionary, 1991). It includes the basic software engineering activities related to requirements engineering, design, implementation, testing, and maintenance, as well as any other activities that result in software products such as software prototyping, software modification, reuse, and system re-engineering (Habib et. al, 2008). There are many different software processes but most of them build on the fundamentals of software engineering, the software lifecycle. The software lifecycle has general representation, consisting of the following main phases (see figure 2.3):

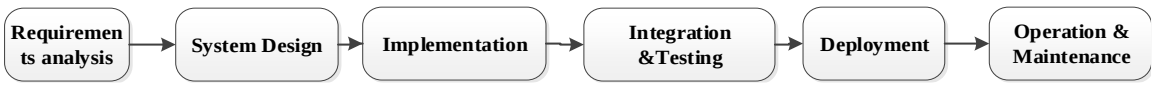


Figure 2.3 A Software Lifecycle (Sommerville, 2011)

Software process contains large amount of activities concerning both engineering and management disciplines (Pfleeger and Atlee, 2006). The main engineering activities concern requirements engineering, software analysis and design, implementation and testing. The main management activities of software process concerns configuration and change management, and project management. The software processes rely on people following them, making decisions and judgments. Since there is no "ideal" software processes, each software organization develops its own software process, defining and structuring their software engineering and management activities in a unique way. The quality, efficiency and composition of software process activities affect the cost of software development, ultimate product quality, and time to market of software product. For this reason, many authors stress the importance of software processes and its quality (Robillard et al., 2003, Sommerville, 2011).

2.2 Software Process Improvement Framework

ISO 9126 (2001) defines quality as "the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs". ISO 25000 (2005) takes the following approach to quality: "capability of software product to satisfy stated and implied needs when used under specified conditions". Assessing software quality early in the development process is essential to identify and allocate resources where they are needed most (Azar et al., 2009). Software process improvements are required to increase the productivity of software companies. It is a systematic approach to improve the capabilities and performance of software organizations. Software Process Improvement (SPI) methodology is defined as a sequence of tasks, tools, and techniques to plan and implement improvement activities to achieve specific goals such as increasing development speed, achieving higher product quality or reducing costs (Sami, 2018). SPI can be considered as process re-engineering or change management project to detect the software development lifecycle inefficiencies and resolve them to have a better process. This process should be mapped and aligned with organizational goals and change drivers to have real value to the organization. Quality in a software product can be improved by process improvement, because there is a correlation between processes and outcomes. According to Sommerville (2011), the software product may be affected by the advancement of the development technology used, the quality of the people involved in producing software product, the process quality, the cost, time and schedule considerations,

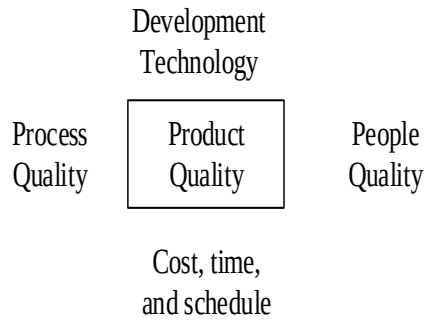


Figure 2.4 Principal product quality factors (Sommerville, 2011)

Process improvement focuses on defining and continually improving process. Defects found in previous efforts are fixed in the next efforts (Jacobs, 2006). The fundamental belief of software process improvement is that improving the process leads to improvements in the final product (Carvalho, 2005). The purpose of improvement is often to enhance software development in order to raise the quality of software. On the other hand, the goal may be to shorten the delivery cycle, to lower the costs and thus improve profitability, or to strengthen the market position. There may also be a need to prove the maturity of development, which many require changes in software development processes (Komi-Sirvio, 2004). An SPI framework defines a set of characteristics that must be present if an effective software process is to be achieved a method for assessing whether those characteristics are present a mechanism for summarizing the results of any assessment, and a strategy for assisting a software organization in implementing those process characteristics that have been found to be weak or missing (Sommerville, 2011). Various researchers designed different models and methodologies for SPI. Capability Maturity Model Integration (CMMI® Product Team, 2002), Software Process Improvement Capability Determination (Eman, 1997), and similar SPI models focus on improving software quality by trying to reduce differences between the process and a standard one. Figure 2.5 below shows the elements of SPI Framework which comprises a set of characteristics that must be present if an effective software process is to be achieved, a method for assessing whether those characteristics are present, a mechanism for summarizing the results for an assessment, and a strategy for assisting a software firm in implementing those process characteristics that have been found to be weak or missing (Pressman, 2009).

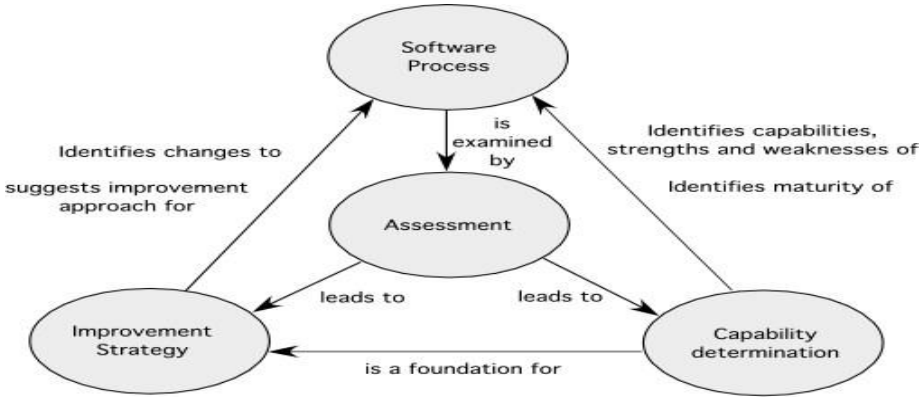


Figure 2.5 Elements of SPI Framework (Pressman, 2009)

According to Pressman (2009) the SPI framework consists of four generic elements which are: software process; assessment; capability determination; and improvement strategy. These elements must be used as a baseline in developing the proposed framework. Thus, there is the need to rearrange these elements to be suitable for software development and improvement issues by integrating the CMMI V2.0 model and DevOps method as generic elements in the developed framework. In this aspect, several changes have done to the contents of the generic elements of SPI framework because the developed software development process improvement framework focuses on development and improvement issues. In this foundation, DevOps has been used as a software development method instead of the improvement strategy, while the CMMI V2.0 model has been used as an improvement model in the developed framework. The basis on which SPI framework for small firms built upon is the step by step improvement as described as shown in the figure 2.6.

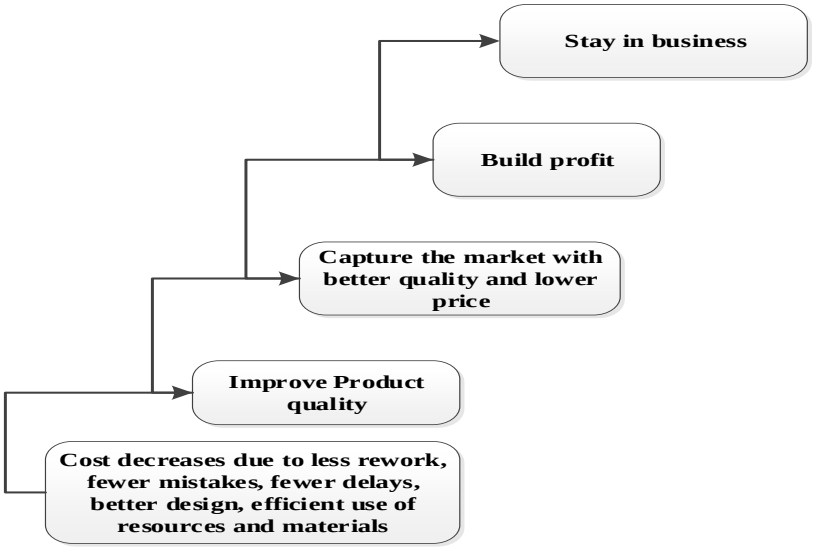


Figure 2.6 The basis of SPI framework for small firms (Kumar and Lal, 2012)

The first two blocks are of major concern in which the SPI framework for small firms is implemented which shows major cost cut offs, less rework, fewer mistakes, fewer delays, better quality products. In the SPI framework Ladder after crossing the first step, the product quality improves. Only after improving quality small scale firms can think of the market with their better-quality products. Building profit is a further step after small scale industries withstand in competition. Profit means they can put more budget and resources to improve the processes effectively. After building profit, small scale firms can go for continuous process improvement with which they can stay in business continuously (Manoj and Sohan 2012).

There are many well-known software process improvement frameworks applicable recently in many software organizations. The following are some of them (Pressman, 2009).

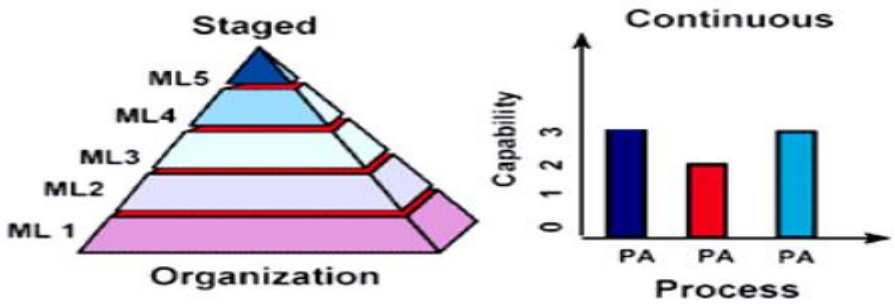
- **Software Process Improvement and Capability dEtermination(SPICE)**
    - An international initiative to support the International Standard ISO/IEC 15504 for (Software) Process Assessment. This framework is only used for assessment purpose. Hence it lacks features to recommend best software practices and validation features.
- **Bootstrap**
    - A SPI framework for small and medium sized organizations that conforms to SPICE
- **Team Software Process (TSP) & Personal Software Process (PSP)**
    - Individual and team specific SPI frameworks that focus on process in-the-small, a more rigorous approach to software development coupled with measurement. It only directed towards team-building and team-working, creating a team environment for establishing and maintaining a self-directed team, and supporting disciplined individual work.
- **TickIT**
    - An auditing method that used only for assessing organizations compliance to ISO Standard 9001:2000.
- **CMMI** –Capability Maturity Model Integration (CMMI product team, 2010) was an initiative of members working in industry, the US government and the SEI, that represents an evolution of CMM models. It is a process model that provides a clear definition of what an organization should do to promote behaviors that lead to improved performance.

2.3 CMMI V2.0

CMMI was developed by a group from industry, government, and the Software Engineering Institute (SEI) at Carnegie Mellon University (SEI, 2010). CMMI models provide guidance for developing or improving processes that meet the business goals of an organization. A CMMI model may also be used as a framework for appraising the process maturity of the organization. By January 2013, the entire CMMI product suite was transferred from the SEI to the CMMI Institute, a newly created organization at Carnegie Mellon. CMMI is the successor of the capability maturity model (CMM) or Software CMM. The CMM was developed from 1987 until 1997. In 2002, version 1.1 was released, version 1.2 followed in August 2006, and version 1.3 in November 2010. Some major changes in CMMI V1.3 are the support of agile software development, improvements to high maturity practices and alignment of the representation (staged and continuous). In March 2018 CMMI 2.0 was introduced.

CMMI is composed of several models and provides best practices to help organizations to improve their processes. These CMMI models provide guidance (best practices) to use when developing processes but they are not processes or process descriptions. They are used for the implementation of any type of product (or system). It is however in the development and maintenance of software that it is most used. Usually, CMMI is the basis for the definition of the software process to be used in the development/maintenance of a specific software system (CMMI product team, 2010). CMMI is currently in version 2.0.

CMMI consists of two representations, as shown in Figure 2.7. These are CMMI continuous representation and CMMI staged representation.



…for a single process area or a set of process areas

…for an established set of process areas across an organization

Figure 2.7 CMMI Staged and Continuous Representation (SEI, 2010)

In **Continuous Representation**, capability improvement of each process area is focused. Organizations can select any number of process areas which are desired to be improved. These process areas then can be improved by implementing the generic goals of the desired capability level. There are four Capability Levels defined in this representation: Incomplete, Performed, Managed and Defined. To achieve a certain capability level, all of the generic goals must be satisfied up to that level.

Staged Representation, on the other hand, focuses on the performance-maturity of an organization as a whole. It defines five Maturity Levels with a number of process areas in each of the upper four levels. These levels are Initial, Managed, Defined, Quantitatively Managed, and Optimizing. To reach a certain maturity level, all the specific and generic goals of the process areas belonging to that level, as well as of the below levels, have to be satisfied. This model is a better option for those organizations who want to make improvements in all processes; want to make sure that all the development processes are performing at the same level; and most of all to reach a certain maturity level for marketing purposes to prove the performance maturity of the organization. Most organizations choose this representation model for its simplicity in ranking the maturity levels from 2 to 5 (Adnet, 2018). Staged model focuses on the overall state of the organization using maturity levels, whereas continuous model focuses on the improvement in the individual process areas using capability levels (SEI, 2010). So that in this study, staged representation of CMMI is chosen for these reasons.

CMMI model components

- ✓ Maturity Levels: indicate process capability
- ✓ Process areas
    - Process areas that are relevant to process capability and improvement are identified.
- ✓ Goals
    - Goals are descriptions of desirable organizational states. Each process area has associated goals.
- ✓ Common Features: implementation/institutionalization criteria
- ✓ Practices
    - Practices are ways of achieving a goal - however, they are advisory and other approaches to achieve the goal may be used.
    - Key Practices are the ‘whats’ (not hows) that must be satisfied to meet the goals of a specific maturity level.

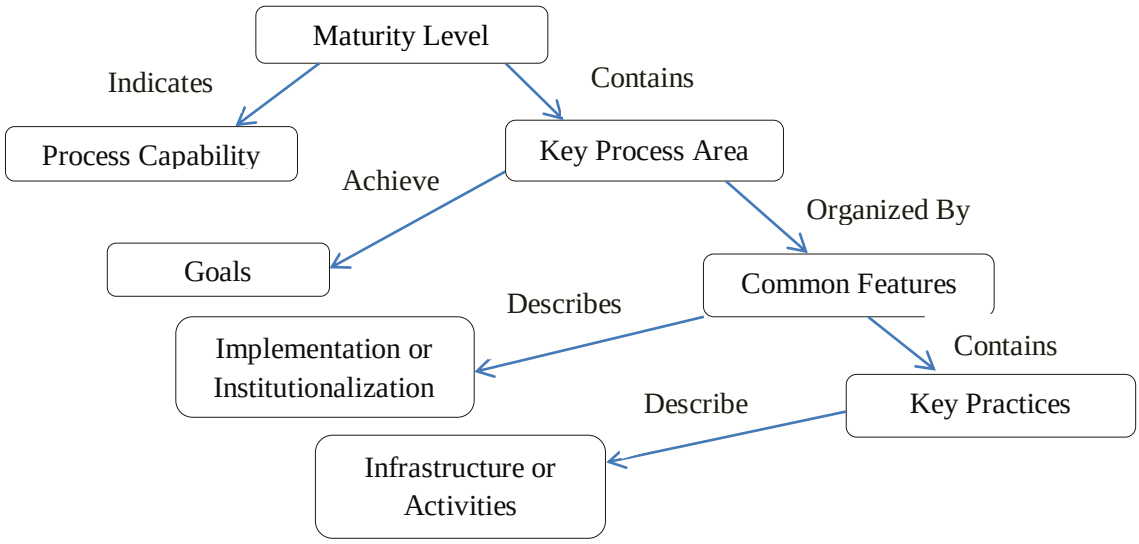


Figure 2.8 CMMI model components (CMMI Product Team, 2010)

Pictorial Representation: An Example

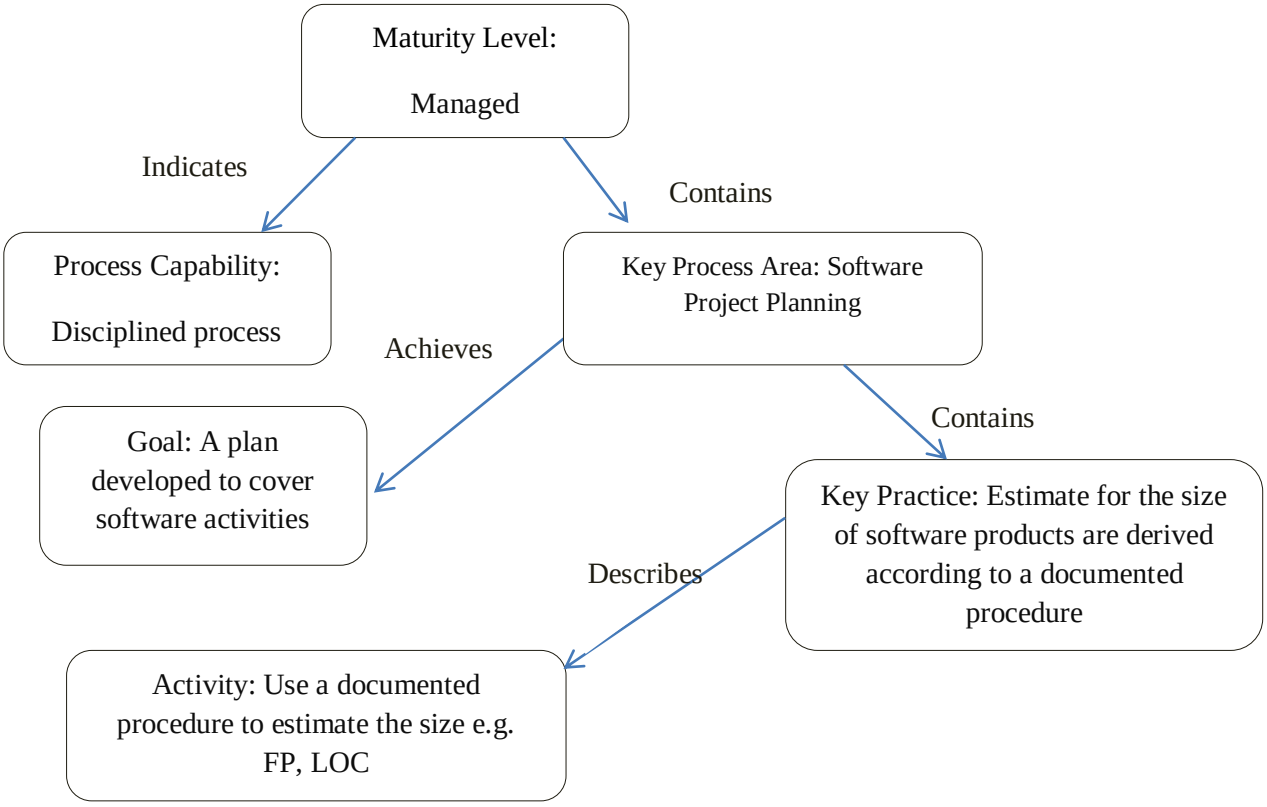


Figure 2.9 CMMI model components Example (CMMI Product Team, 2010)

Table 2.1 lists the four capability levels and the five maturity levels of CMMI version 2.0. It can be noticed that both representations use the same names for their 2nd and 3rd levels, i.e., Managed and Defined. In staged model, there is no maturity level 0, whereas in continuous model, there do not exist capability levels 4 and 5. Also, the names of the maturity level 1 and capability level 1 are different (SEI, 2010).

Table 2.1 Maturity Levels vs. Capability Levels (SEI, 2010).

Level	**Continuous Representation Capability Levels**	**Staged Representation Maturity Levels**
Level **0**	Incomplete	___
Level **1**	Performed	Initial
Level **2**	Managed	Managed
Level **3**	Defined	Defined
Level **4**	___	Quantitatively Managed
Level **5**	___	Optimizing

Maturity levels help in making improvement across multiple process areas in an organization and assign the organization an overall maturity rating. Each maturity level contains a predefined set of process areas which are improved and matured before going to the next maturity level processes. An organization can reach a certain maturity level by satisfying the specific and generic goals of the process areas of that level. There are five maturity levels defined in CMMI (see figure 2.10), numbered from 1 to 5. A short description of each of the levels is given below. This description is taken from the CMMI-DEV version 2.0 document published by CMMI Institute (CMMI Institute Partner Workshop, 2019).

1. **Initial** – At maturity level 1, processes are usually ad hoc and chaotic. The organization usually does not provide a stable environment to support processes. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes. In spite of this chaos, maturity level 1 organizations often produce products and services that work, but they frequently exceed the budget and schedule documented in their plans.
2. **Managed** – At maturity level 2, an organization has achieved all specific and generic goals of the maturity level 2 process areas. At maturity level 2, the projects have ensured that processes are planned and executed in accordance with policy; the projects employ skilled

people who have adequate resources to produce controlled outputs; involve relevant stakeholders; are monitored, controlled, and reviewed; and are evaluated for adherence to their process descriptions.

3. **Defined** – At maturity level 3, an organization has achieved all the specific and generic goals of the process areas assigned to maturity levels 2 and 3. At maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and method. Processes are tailored from the organization's set of standard processes to suit a particular project or organizational unit…The organization's set of standard processes, which is the basis for maturity level 3, is established and improved over time.
4. **Quantitatively Managed** – At maturity level 4, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, and 4 and the generic goals assigned to maturity levels 2 and 3. At maturity level 4, the organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing projects. At maturity level 4, the performance of projects and selected sub processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable; whereas at maturity level 3, processes are only qualitatively predictable.
5. **Optimizing** – At maturity level 5, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, 4, and 5 and the generic goals assigned to maturity levels 2 and 3. At maturity level 5, an organization continually improves its processes based on a quantitative understanding of its business objectives and performance needs. Maturity level 5 focuses on continually improving process performance through incremental and innovative process and technological improvements. A critical distinction between maturity levels 4 and 5 is the focus on managing and improving organizational performance. At maturity level 4, the organization and projects focus on understanding and controlling performance at the sub process level and using the results to manage projects. At maturity level 5, the organization is concerned with overall organizational performance using data collected from multiple projects (SEI, 2010).

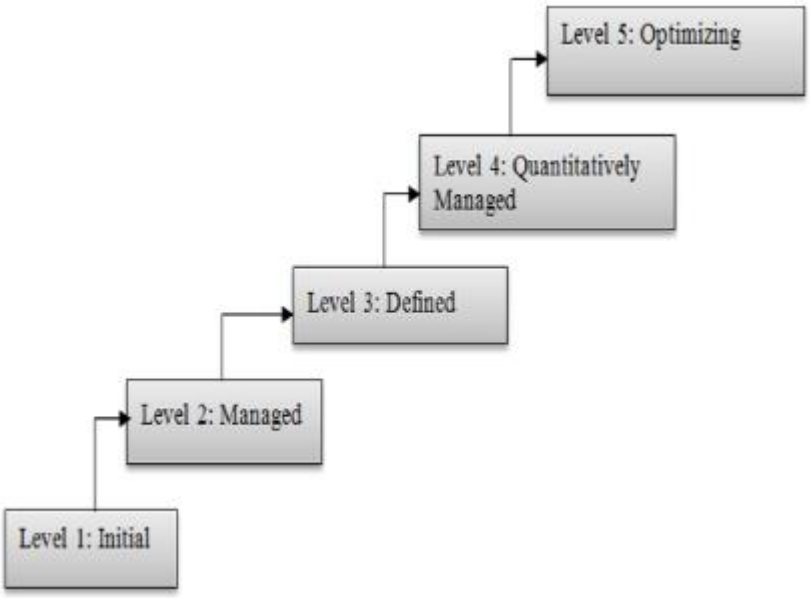


Figure 2.10 CMMI V2.0 maturity levels (SEI, 2010)

Table 2.2 CMMI Levels of Maturity (CMMI Institute, 2019b)

Maturity Level	**1- Very Low Initial**	**2- Low Managed**	**3- Medium Defined**	**4- High Quantitatively Managed**	**5- Very High Optimized**
characteristics	Ad-hoc, informal, not implemented, unpredictable, reactive.	Implemented only in a single team, works well in a project, often reactive	Implemented within an entire organization, standardized	Widely deployed, processes measured and controlled, benchmarked	Continuously improved, root cause analysis, resolution

According to CMMI Institute (2019a) CMMI-DEV version 2.0 addresses 18 practice areas plus safety/ security practices for process improvement and evaluation. Each practice area is included under capability areas, and each capability area is included under four categories.

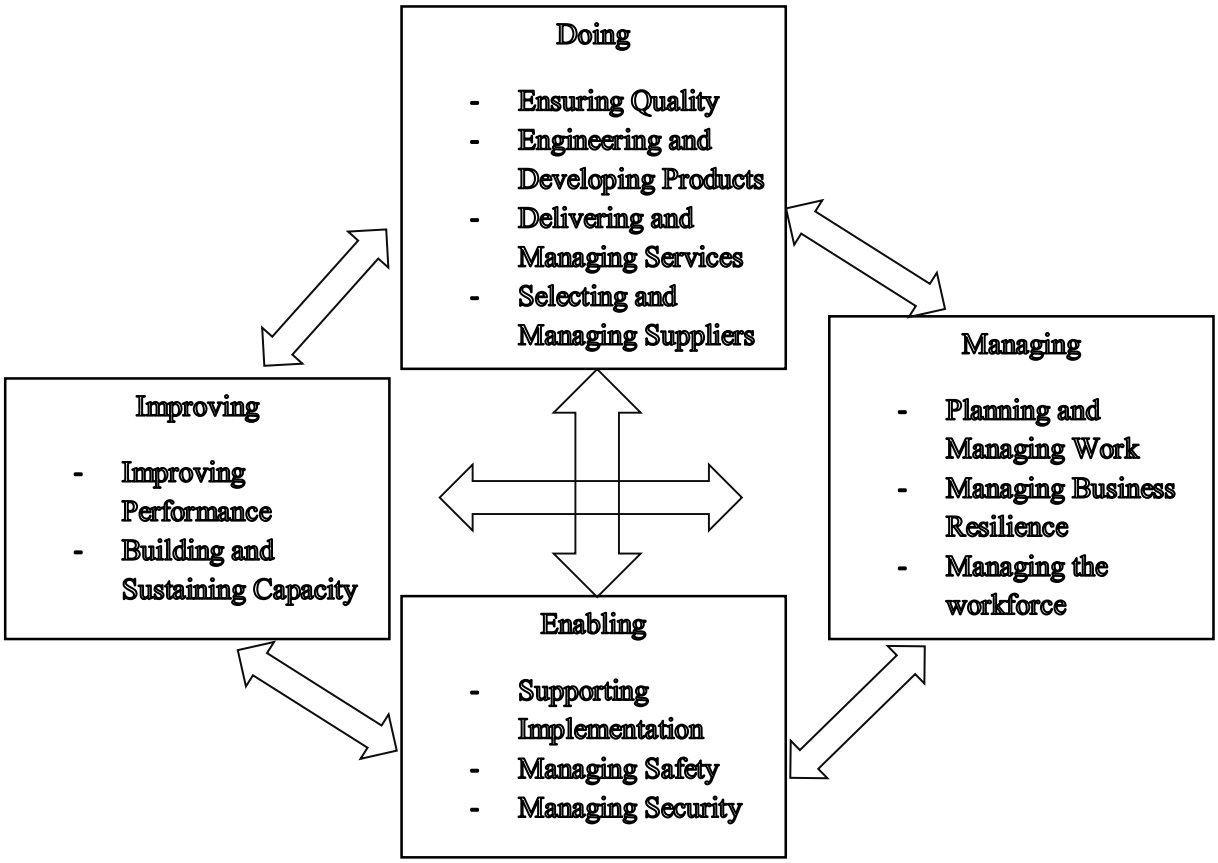


Figure 2.11 CMMI Dev 2.0 capability areas (CMMI Institute, 2019a)

Hereunder table 2.3 shows the "Categories", "Capability Areas", and “Practice Areas" of CMMI V2.0.

Table 2.3 Capability Areas within a Category (CMMI Institute Partner Workshop, 2019)

Category	**Capability Area**	**Practice Area**
Doing	Ensuring Quality	Requirements Development and Management
		Process Quality Assurance
		Verification and Validation
		Peer Review
	Engineering and Developing Products	Technical Solution
		Product Integration
	Selecting and Managing Suppliers	Supplier Agreement Management
Managing	Planning and Managing Work	Estimating
		Planning
		Monitor and Control
	Managing Business Resilience	Risk and Opportunity Management
	Managing the Workforce	Organizational Training
Enabling	Supporting Implementation	Casual Analysis and Training
		Decision Analysis and Training
		Configuration Management
Improving	Building and Sustaining Capacity	Governance
		Implementation Infrastructure
	Improving Performance	Process Management
		Process Asset Development
		Managing Performance and Measurement

Practice areas descriptions (CMMI Institute Partner Workshop, 2019)

- ✓ Requirements Development and Management: is applied to elicit, analyze, and establish customer, product, and product component requirements.
- ✓ Process Quality Assurance: provides staff and management with objective insight into processes and associated work products.
- ✓ Verification and Validation: Validation demonstrates that a product or product component fulfills its intended use when placed in its intended environment. Verification (VER) ensures that selected work products meet their specified requirements.
- ✓ Peer Review: its purpose is used to remove defects from the software work products early and efficiently
- ✓ Technical Support: a practice to select design and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product related lifecycle processes either singly or in combination as appropriate.
- ✓ Product Integration: is used to assemble the product from the product components, ensure that the product, as integrated, behaves properly (i.e., possesses the required functionality and quality attributes), and deliver the product.
- ✓ Supplier Agreement Management: manage the acquisition of products from suppliers.
- ✓ Estimating: Project planning parameters include all information needed by the project to perform the necessary planning, organizing, staffing, directing, coordinating, reporting, and budgeting are established and maintained
- ✓ Planning: to establish and maintain plans that defines project activities.
- ✓ Monitor and Control: is applied to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan.
- ✓ Risk and Opportunity Management: identify potential problems before they occur so that risk handling activities can be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives.
- ✓ Organizational Training: is used to develop skills and knowledge of people so they can perform their roles effectively and efficiently.
- ✓ Casual Analysis and Training: is used to identify causes of selected outcomes and take action to improve process performance
- ✓ Decision Analysis and Training: is used to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria.

- ✓ Configuration Management: is used to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.
- ✓ Governance: makes sure that senior management actively leads the way
- ✓ Implementation Infrastructure: makes sure processes are performed by providing sufficient resources, funding, and training for developing processes
- ✓ Process Management: contain the cross-project activities related to defining, planning, deploying, implementing, monitoring, controlling, appraising, measuring, and improving processes
- ✓ Process Asset Development: is a systematic process of developing, operating, maintaining, upgrading, and disposing of assets in the most cost-effective manner (including all costs, risks and performance attributes).
- ✓ Managing Performance and Measurement: proactively manage the organization's performance to meet its business objectives.

Table 2.4 New and Most Significantly Changed Practice Areas under CMMI V2.0 from CMMI V1.3 (CMMI Institute Partner Workshop, 2019)

Types of Change	**Practice Area**
New Practice	Estimating
	Peer Reviews
	Governance
	Implementation Infrastructure
Most Significantly Changed Practice Areas	Managing Performance and Measurement
	Risk and Opportunity Management
	Verification and Validation
	Requirements Development and Management

The CMMI, which was developed by the SEI of Carnegie Mellon University, was selected as the basis on which to develop a more tailored SPI framework suitable for small organizations. The CMMI consists of a collection of recommended practices that provide guidance for process improvement in developing software products and services, managing resources, guiding training programs, appraising process methods, and acquiring IS products and services (SEI, 2010). It is the leading reference model for process improvement (Niazi et al., 2010). It achieved wide acceptance

in the software industry and is used by many firms worldwide (Staples, &Niazi, 2008; Jiang et al., 2004) to increase the likelihood of producing higher quality software (Agrawal, & Chari, 2007). This model is chosen as the focus of this research since CMMI based assessments integrated with other assessments are a widely used approach for evaluating the software processes within a company (Trudel et al., 2006) and indicating its key weaknesses for immediate attention and improvement (Daskalantona, 1994). Secondly, the beneficial experiences of CMMI programs have been reported as a part of many studies during the past decades (Galin and Avrahami, 2006; Niazi et al., 2003; Stelzer and Mellis, 1998). For example, the results of CMM programs in 400 projects report improvements in productivity and development speeds in software development due to the CMMI programs (Galin and Avrahami, 2006; Stelzer and Mellis, 1998). Based on the results of the analysis, they report a 26 to 187% improvement in productivity, a 28-53% improvement in cycle time and a 120-650% percent return in investment due to the use of CMMI programs (Galin and Avrahami, 2006). For these reasons, the CMMI was chosen as the basis for developing a simplified SPI framework for use by small software firms.

The recent version of CMMI, i.e. CMMI V2.0, is selected among the previous versions because of the key improvements made. It changes “Process Area” to “Practice Area” so that it emphasizes that the model is not a collection of (rote) processes to be implemented, but a collection of practices to be used to run and manage projects. Performance capabilities are incorporated in every level to help organizations identify performance needs, establish performance goals, and track, measure, and achieve performance goals. It also improves value of CMMI appraisals by lowering time, effort, and cost. The recent version keep CMMI current with latest methodologies like scalable architecture includes additional method guidance, hosted online so more readily updated, new content additions, such as Safety and Security, that address critical business needs and make CMMI V2.0 more users friendly. Non-technical / Plain Language makes it easier for users to read and understand the model. Online platform allows users to customize and tailor the model to fit specific organizational needs and it supports multiple languages like English, Spanish, and French. CMMI V2.0 helps organizations quickly leverage the key capabilities that directly impact the ability to drive business results, higher quality, and focus on performance, reduction in cost, time to market and risk (CMMI Institute, 2019b).

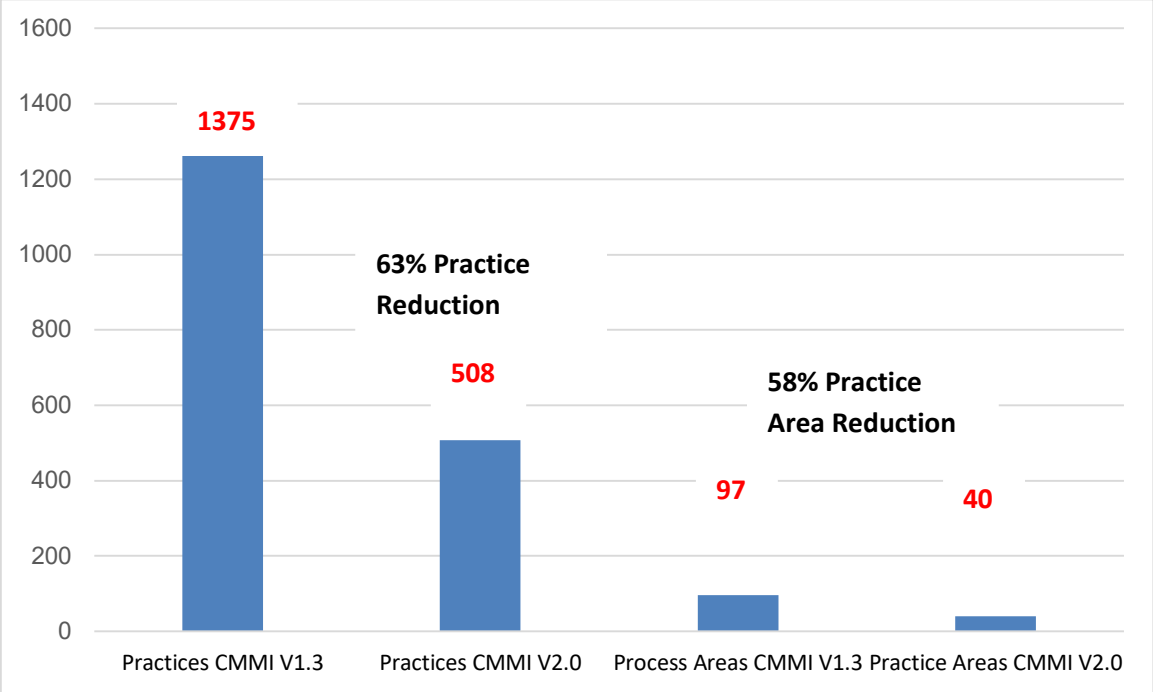


Figure 2.12 CMMI V1.3 vs. CMMI V2.0 by the numbers (CMMI Institute, 2019a)

From figure 2.12, we can easily notice that it is advantageous to use CMMI V2.0 because there is a reduction in both practices and practice areas from version 1.3 to version 2.0. There exists 63% practice reduction and 58% practice area reduction when we use the recent version, CMMI V2.0, which saves time and cost needed for implementing the software process improvement initiative. (CMMI Institute, 2019a).

2.4 DevOps Approach

Due to a wide variety of organizational profiles and specifics of the products produced by different software companies, it is impossible to have one "ideal" prescribed software process that would fit all. This, however, does not mean that the software companies have no guidelines on how to structure their software processes. Instead, the organizations can choose from a wide range of different software process models, and software development methods that are designed to guide towards suitable and efficient software processes. Software process models and methods started to emerge in early 1970s with the Waterfall model. Hundreds of other models and methods have been created since then. However, only some of them have become widely accepted today. Among these are Waterfall model, V-Model, Spiral model, Iterative model, Rational unified process, Model driven development, Agile development, Test driven development, Crystal, Scrum, XP, Lean, Kanban, etc. The Waterfall model is the oldest and the traditional approach to software development in which development happens in a step by step approach, while agile software development presents the

modern trend of development. Some call DevOps the 3rd generation software development method, a continuation from the 2nd generation agile methods (Eficode, 2016).

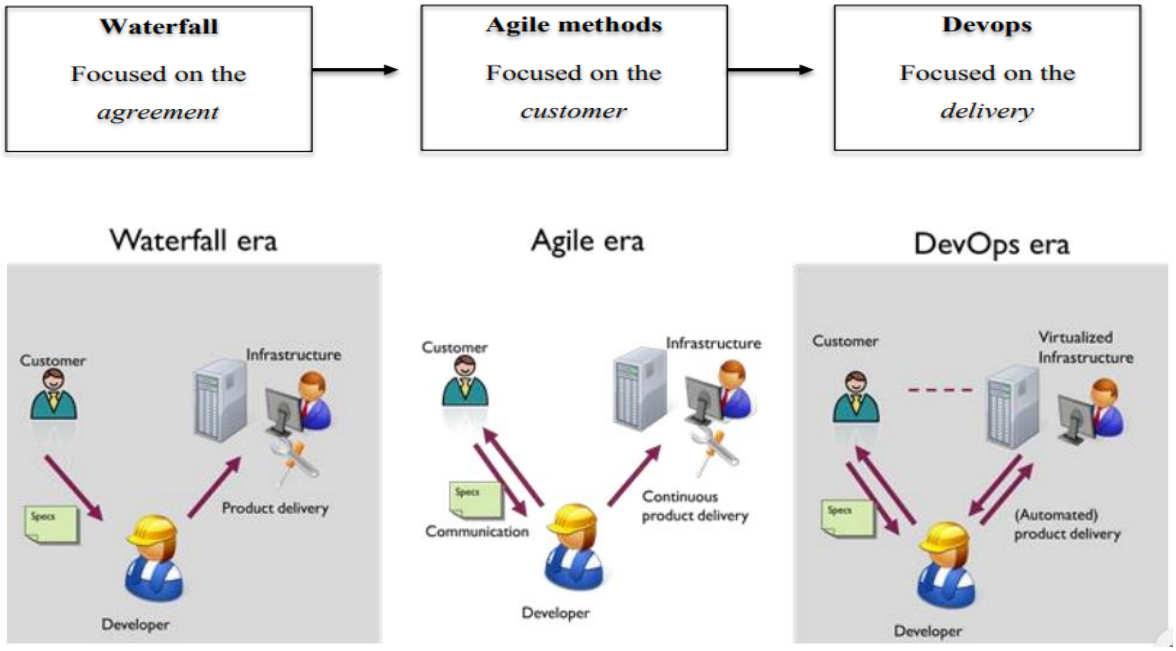


Figure 2.13 DevOps: 3rd generation software development method (Eficode, 2016)

In waterfall model, first programmers accept the client requirements and analyze it, programmers then come up with a project plan and the design architecture. Third, programmers code the application according to the plan and the design. Fourth testing ensures the application error free and meets the requirements. Finally, the application is delivered to the client; after the application is delivered the operation team maintains the application. Any new requirement from the user will restart the development cycle. If the client is unhappy with the product delivered, the entire project cycle is restarted. If the clients' list of changes contains unclear requirements the project cannot start and eventually delayed (Eficode, 2016).

Following the agile model, programmers create prototypes to understand client requirements. First the client sent his/her requirements to the programmer, then the programmer creates a prototype of the application sent it to the client, the client provides feedback, and list of changes to be made to the programmer. The entire process of building software is broken down into small actionable blocks called sprints that contain four phases as plan, code, test, and review (Eficode, 2016). Here the client's requirements better understood because of constant feedback, and the product is delivered much faster as compared to the waterfall model. However, the product gets tested only on the developer computers not on production systems. In addition, developers and developer teams work

in silos. In agile development model, the developer submits the product to operations team for deployment. When the product fails in the production system, the operation team is clueless and sends the product back to the product team. This is the main reason why many companies try to adopt DevOps.

Among the above sort of software development methodologies, DevOps is used in this thesis. DevOps is a relatively new term that first appeared in 2009. It is an evolution from agile software development model. Agile addressed the gap between clients and developers. While DevOps addresses the gap between developers and operations. Development team will submit the application to the operations team for implementation. Operations team will monitor the application and provide relevant feedback to developers. DevOps is chosen because of its benefit. Several researchers have claimed that DevOps has no clear definition (Lwakatare, 2017; Jabbari et al., 2016; Roche, 2013) and so several interpretations of it are observed among practitioners (Erich et al., 2017). In current literatures, multiple scholars are attempting to bring up suitable definitions for DevOps. Jabbari et al. (2016) performed a systematic literature study to find out the most occurring components within DevOps definitions found in the literature studies available until 2016. They identified that communication, collaboration and team working, 'bridge the gap', development method, software delivery, automated deployment, continuous integration, and quality assurance are key terms next to development and operations. According to Anon (2018) DevOps is described by many terms as shown in figure 2.14.

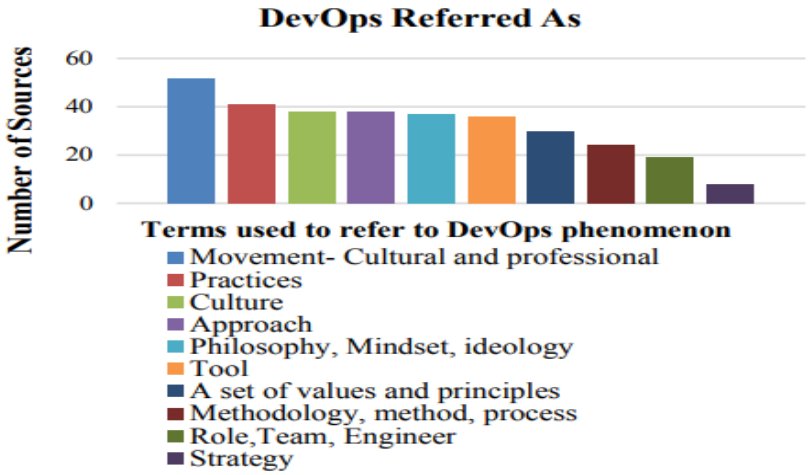


Figure 2.14 Terms to refer DevOps (Anon, 2018)

Thus, it can be clarified that DevOps is not restricted only to Developers and Operations team members of an organization but also to other extended team roles such as architects and product owners (Swartout, 2014). Therefore, how we grow DevOps culture and practices in our organizations needs more attention (Kromhout, 2017). DevOps aims to break down the silos between inter departments and allowing everyone involved in the process of delivering a product such as system engineers, DBs and network engineers, security to communicate and share their concerns. DevOps is not only a set of tools and procedures to follow; it is a culture for IT departments (Wieringa, 2014). On the other hand, measurement is an important aspect of DevOps, and it is important to improve the product because if a product is measurable then it is possible to improve it (Wieringa, 2014).

In addition to Wieringa (2014), Humble and Molesky (2011) have also proposed that DevOps consists of four dimensions: Culture, Automation, Measurement and Sharing. DevOps means a culture shift toward collaboration between development, quality assurance, and operations (Ebert et al., 2016), or DevOps is about aligning the incentives of everybody involved in delivering software (Humble and Molesky, 2011) where its success is based on four practices:

- ✓ **Culture**: joint responsibility for the delivery of high-quality software. It is represented by human communication, technical processes, and tools. DevOps aims to break down the silos between inter departments and allowing everyone involved in the process of delivering a product such as system engineers, DBs and network engineers, security to communicate and share their concerns (UDACITY, (2016a).
- ✓ **Automation**: automation of processes in all development and operation steps towards rapid delivery and feedback from users. Many people focus on the productivity gains (output per worker per hour) as the main reason to adopt DevOps. Not only automation used to save time, but also to prevent defects, create consistency, and enable self-service (UDACITY, (2016b).
- ✓ **Measurement**: all process must be quantified to understand delivery capability and setting goals to improve the process. DevOps supports continuous delivery and deployment. Continuous delivery requires continuous improvement of the product; thus, DevOps supports measurement of KPIs and makes it one of its core values because the movement believes it is difficult to improve without the ability to measure. Decisions based on visible and easy-to-read data are the key to having the right choices. The data should be available, transparent, accessible, related visually (UDACITY, (2016c).

- ✓ **Sharing**: it is crucial the sharing of feedback, best practices, and knowledge enabled by tools. DevOps realized the power of sharing and the significant impact it brings. Within the organization sharing tools, findings, defects, and experiences enable the individuals who share similar interests and needs to meet. Such a move makes the process of finding new opportunities to collaboration much easier, eliminating duplicated work along with having a powerful sense of commitment. DevOps also, promote sharing resources (code, documentation… etc.) outside the organization with the related community. It helps the organization to find new features, find defects and energize the individuals (UDACITY, (2016d).

According to Anon (2018), the workflow in software development and delivery is divided into eight phases. DevOps requires a delivery cycle that comprises planning, development, testing, deployment, release, and monitoring with active cooperation and real time collaboration between different members of a team. In plan stage, business owners and software development team discuss project goals and create a plan, programmers then design and code the application and use tools like Git to store application code, build tools like Maven and Gradle (Ebbers, 2016) take code from different repositories and combine them to build the complete application, then the application is tested using automation testing tools like Selenium and JUnit to ensure software quality. When testing is complete, new features are integrated automatically to the existing codebase. Application is packaged after release and deployed from deployment server to production server. Once the software is deployed, operation teams perform activities such as configuring servers and provisioning them with the required resources. Finally, monitoring the entire environment to identify specific issues of specific release and understand the impact on end users. Description of DevOps phases are presented in figure 2.15 below.

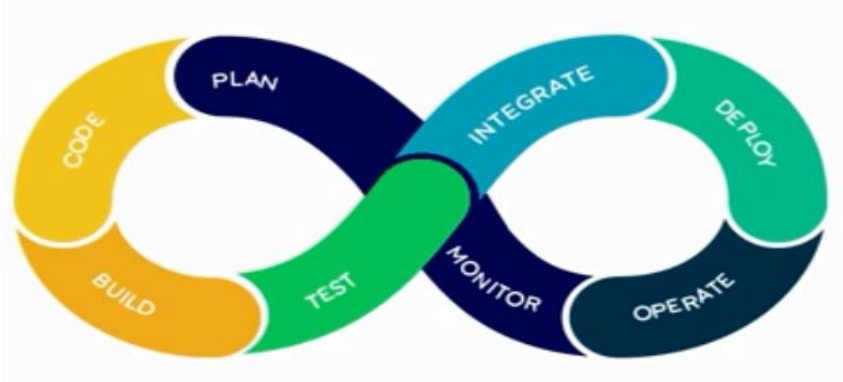


Figure 2.15 DevOps Phases (Anon, 2018)

DevOps contains three processes. These are continuous delivery, continuous integration, and continuous deployment. Continuous deployment is an industrial paradigm characterized by the frequent, rapid and automated release of software changes, including new features, to customers as soon as the changes are committed by software developer so as to gain quick feedback (Humble & Farley 2010). In contrast, continuous delivery ensures that software changes are developed rapidly but not released immediately to the customers after software developer's code commit. Automated deployment is the ability to get software deployed in any environment at any given time (Olsson et al., 2012).

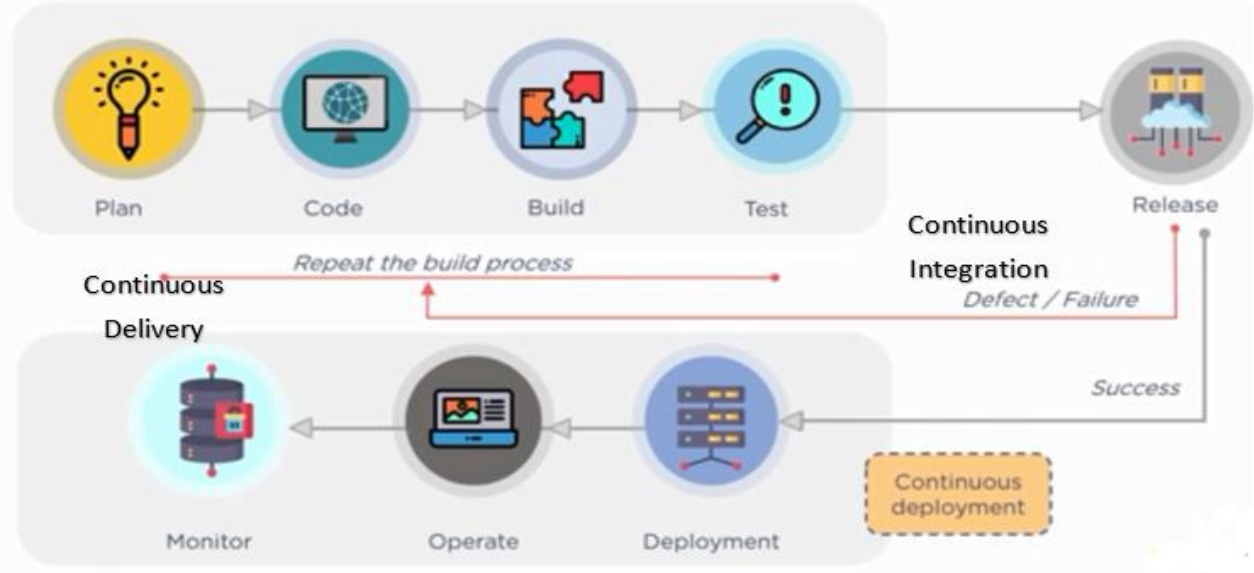


Figure 2.16 DevOps Processes (Lwakatare, 2017)

According to Puppet Labs (2015) State of DevOps report, by minimizing the bottlenecks and reducing the pains of deployment, high performing organizations deploy 30 times more frequently than low-performing ones. In this context, high-performing organizations use DevOps approaches. More deploys means faster time-to-market and an enhanced feedback loop. About 20 minutes deployment to production, depending on exploratory tests and upload time to cloud (Elberzhager et al, 2017) and 1.1 days average release cycle time, an 85% reduction in release cycle time (Callanan and Spillane, 2016) reduced release cycle was obtained. The 2015 State of DevOps research states high-performing organizations have 60 times higher rate of success in deployments than low-performing ones (Puppet Labs, 2015).According to Chandu (2019)analysis on DevOps Zone, there are five reasons on why firms adopt DevOps method. These are shorter development cycles, faster innovation, reduce implementation failure, reflections and recovery time, better communication and cooperation, greater competencies, and reduce costs and IT staff. All the benefits of DevOps translate

into reduced general costs and requirements of IT staff. DevOps development teams require IT staff to be 35 percent less and IT costs 30 percent lower.

2.5 Small Scale Software Firms

A small-scale enterprise, or more simply, a small business, is one marked by a limited number of employees and a limited flow of finances and materials (Sarokin, 2019). Most of the software development organizations all over the world are small software development firms. These firms play a crucial role in the economy of many countries, where they develop a large portion of the required software applications, offer many job opportunities, and exploit new technologies (Savolainen et al., 2007). These firms have realized that it is necessary to organize and improve their software development and management activities. Traditional software process improvement (SPI) models and standards are generally not possible to be implemented directly by small firms, as these firms are not capable of investing the cost of implementing these programs due to limited resources and strict deadlines to complete the projects. In addition, the existing regional SPI models which were developed for small firms are not suitable to be implemented by these firms all over the world. Furthermore, small firms also have ignored the software development practices explaining "how to do the improvement"; as they only focus on "what to do for improvement".

Even if it is said that there is no single cutoff to distinguish small from medium or large businesses, different government agencies and private organizations rely on different sets of criteria when talking about a small firm (Sarokin, 2019)**.** According to Sarokin, (2019), in some cases, industry sector matters as well. A 250-person company in the automobile industry, say, may be considered small while a similar sized firm might be categorized as a medium-scale operation in another sector, such as clothing manufacture or legal services. So do with annual revenue. A company doing$5 million a year in sales might be labeled a small business in some contexts but not in others.

Prior studies of small software development firms, whether standalone software business or software organizations within a larger business, identified as having 70 or fewer software employees. The software industry, however, is largely made up of smaller organizations with fewer than 20 software employees (Brodman and Johnson, 1994; Goldenson and Herbsleb, 1995). Furthermore, small organizations are defined by the ABS (2002) as businesses employing less than 20 people. The ABS recognizes three categories within that definition: non-employing businesses (sole-proprietorships and partnerships without employees); businesses with one to four employees; and businesses with between five and 19 employees. The first two categories are sometimes referred to as micro-

businesses. Organizations with 20-199 employees are classed as medium sized businesses. This study conforms to the ABS definition of small firms that may run micro-businesses. Like their larger counterparts, these small software firms also compete in a global market. However, to win these global contracts, they must reveal that their software development processes are sufficiently capable and mature (Niazi et al., 2010; Turner Consulting Group, 2007). Software quality is therefore important not only to address customer demand, but also as the gateway for enabling firms to compete effectively in a global market.

Studies have shown that many small software firms are often not aware of SPI programs and their benefits (Niazi et al., 2010); and if they are, they do not adopt SPI initiatives. In addition, small software firms are typically characterized by informal processes that emphasize getting the software out in order to survive, limited staff to develop specialized functions in software development, and lightweight processes with limited budgets to perform process improvement. They also typically have flat structures with the Chief Executive Officer (CEO) making most of the decisions (Paulk, 1998; Callanan & Spillane, 2016; Scott et al., 2001), and project managers performing many functions in the implementation of IS projects. These factors, coupled with the resource constraints experienced by small firms, will normally prohibit the adoption of the full-blown CMMI framework. Ethiopia is one of the fastest developing economies in the world. Small firms are major players in the economic resurgence of the country. They are instrument of change and vehicles of growth and diversification. These small firms contribute for almost 3.4% to GDP and 90% to employment (Central Statistics Authority, 2003). However, Ethiopia ranked very low, at 169th out of 175 countries, on the 2016 ICT Development Index (IDI).The IDI is one of the indices that measures ICT readiness using three sub-indices: infrastructure and access, use, and skills. Ethiopia's standing was well below Mozambique, Tanzania, and Malawi, countries that also hit low in the IDI (Solomon, 2017). Generally, why small-scale software firms are an important issue and the simple answer is the economic benefits obtained from these firms.

2.6 Related Works

To have conceptual understanding and recognize the breach that is revealed by prior studies diverse materials, including journal articles, conference papers, and books are appraised. Several studies related to frameworks for software process improvement have identified. Numerous initiatives have presented approaches, process and tools to support small firms worldwide. The ISO/IEC JTC1 SC7 Working Group 24 recognized that the existing standards ISO/IEC 12207 and ISO/IEC 15504 do not explicitly address the needs of small firms, and developed a set of guidelines and profiles for firms with fewer than 25 employees (ISO/IEC 29110: Software Engineering - Lifecycle Profiles for Very Small Entities), expected for release in 2010 (Laporte et al, 2008b). From the CMMI side, the SEI hosted in 2005, an international workshop to provide insight and guidance about process improvement in small settings, and later released (2008) a field guide with step-by-step guidelines and examples for small firms. Several other initiatives -based on the most conspicuous models, from ISO/IEC and SEI have intended to help small firms by either defining assessment methods personalized to their context and/or recommending preselected sets of processes for small firms (Staples and Niazi, 2008; Pino et al., 2010). These include RAPID (Rapid Assessment for Process Improvement for software Development) (McCaffery, 2007) and Adept (Cater-Steel et al., 2006). Other proposals focus on adapting and generating models for SPI in small settings, including Impact (Scott et al., 2001) and IDEAL adaptations, (Casey and Richardson, 2004; Pino et al., 2010) includes a comprehensive analysis of such initiatives.

In Latin America, MoProSoft (Oktaba, 2006) combined and adapted to Mexican settings the recommended practices of the now-retired CMM v1.1 (levels 2 and 3), ISO 9001:2000, and other specialized models such as Project Management Body of Knowledge (PMBOK). It also offers an associated assessment method called EvalProSoft (Process Assessment Method for Software Industry), based on ISO/IEC 15504. In Brazil, the MPS.BR proposal (Weber et al., 2005) also considered several international standards and adapted them to the Brazilian reality. Its main products are a reference process called MR-MPS (Modelo de Referência para Melhoria de Processo de Software) and the evaluation method MA-MPS (Método de Avaliação para Melhoria de Processo de Software). These products are conformant with ISO/IEC 15504 and ISO/IEC 12207, and compatible with CMMI. Another Brazilian initiative is the MARES (Método de Avaliação de pRocEsso de Software) (Wangenheim, 2006) assessment model for small firms. The CYTED-sponsored Iberoamerican project Competisoft (Oktaba et al., 2007) borrowed heavily from several

other Latin American models and initiatives, mainly from MoProSoft and EvalProSoft; it also incorporates a process improvement model, called PmCompetisoft, based on the IDEAL model and practices of Extreme Programming and Scrum, to enable a lightweight approach to process improvement. There are also some interesting support software tools. The Taba Workstation is a process-centered software engineering environment, composed of several integrated CASE tools that allow organizations to define, deploy and improve their processes (Ferreira et al., 2006); it has been used to support the implementation of MPS.BR-based SPI initiatives in Brazilian small settings (Montoni et al., 2007). Also, two tools complement the Competisoft project: (Genesis Pino et al., 2009) supports implementation of an SPI initiative and the administration of generated knowledge. And the Ramala tool (Rimawi et al., 2005), which supports process assessment, process definition, and process improvement tracking, based on CMMI, ISO/IEC 15504 and PMBOK.

A study by Khan et al. (2010) proposed a CMM framework for software process improvement for small and medium enterprises (SME). The framework is based on agile methodology for Pakistan software industry for successful software projects. Even if it covers all software engineering practices in the 5 CMMI levels and successfully mapping 305 KPAs with XP practices, the framework focuses on software engineering practices only and no clear guidelines to implement these practices. It also has no real-life evaluation. Lukasiewicz and Miler (2012) conducted a study to build a framework based on CMM and Scrum model to improve agility and discipline of software development which covers most of the project management aspects in CMMI level 2 and 3 which is 60% of these 2 levels KPAs and the model was evaluated with good results. However, they focused on project management practices only. It doesn't cover level 4 and 5 KPAs and CMMI level 3 KPAs related to organizational aspects are not included.

Habib et al. (2008) also conducted a study to offer a framework by combining six sigma and CMMI-an approach to rush process improvement in small and medium sized enterprises. The authors successfully mix CMMI and six-sigma, but it also focuses on project management practices only and don't cover CMMI level 4 and 5 KPAs. Correspondingly, Zhang and Shao (2011) conducted a study that proposes a software process improvement framework for SMEs based on CMMI. They fully cover KPAs for CMMI level 2 and 3 and give the required awareness for hastening organizational improvement. But they don't take any agile methods benefits. Having the above mentioned problems including addressing as much KPAs as possible and recommending on how to perform the improvement program on small firm settings, the framework integrating CMMI V2.0 and DevOps method have used.

CHAPTER THREE

METHODOLOGY

3.1 Overview

This part of the thesis deals with research methodology. A methodology is "a system of principles, practices, and procedures applied to a specific branch of knowledge to obtain better performance by using scientific methods of gathering and interpreting information. (Eekels and Roozenburg, 1991). The basic research processes include problem discovery, problem definition, research method selection, sample design, data collection, data encoding, data processing and analysis, result interpretation and reporting.

3.2 Research Design and Approach

To fulfill the aims of this research, design science research methodology (Rossi. and Sein, 2001; March and Smith, 1995; Sein et. al., 2011) was adopted as research methodology because it is of importance in a discipline oriented to the creation of successful artifacts. The design science process includes six steps: problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication (Peffers et. al., 2007). Literatures have been reviewed to understand the state of the problem (problem identification), the objectives of a solution have identified, the design of a software process improvement framework have explicated, the use of the framework was demonstrated in three small firms (demonstration), and lastly, the framework in terms of the objectives of the study was evaluated (evaluation) and concluded. The research by its nature needs the researcher to be embedded in the company for the duration of the thesis. The project work and the software process improvement effort were observed.

Data were gathered by interviewing members of the firms, questionnaire, observing the work done, and by analyzing work artifacts and other available documents. Standard questionnaires were used to assess the maturity level of the organization through selected employees of the organization. It was conducted to assess the process maturity level which helps as the starting benchmark to measure the performance of the firm. Assessing the maturity level of the firm helps to recommend future process improvement efforts. Formal assessment was used to assess whether a firm is at a certain maturity level. Formal assessment is an appraisal aimed to determine the maturity level of an organization based on its current condition, to provide prioritization on the organization's issues, and to gain commitment/support for software process improvement (Herbsleb et al, 1997).

DevOps method was integrated with CMMI V2.0 best practices. DevOps is a software development methodology that combines software development (Dev) with information technology operations (Ops) which was published in 2018 (Anon, 2018). The goal of DevOps is to shorten the system development life cycle while also delivering features, fixes, and updates frequently in close alignment with business objectives. In the DevOps, developers receive fast, constant feedback on their work, which enables them to quickly and independently implement, integrate, and validate their code, and have the code deployed into the production environment (deploying the code may or may not be done by themselves) (Lwakatare, 2017). CMMI V2.0 is a globally recognized set of best practices that enable organizations to improve performance, key capabilities, and critical business processes. Currently, CMMI V2.0 includes specific guidance to help firms that use agile methods for development to strengthen their processes and scale their agile practices with a focus on performance. Complete assessment is a detailed task that requires organization wide participation. It consists of the phases: Select a team, complete the maturity questionnaire, analyze the responses, conduct a detailed site visit, produce a list of findings of strengths and weaknesses, and prepare a key process area profile of the organization.

This study adapted design science process model presented in Peffers etal. (2007), as shown in figure 3.1 below.

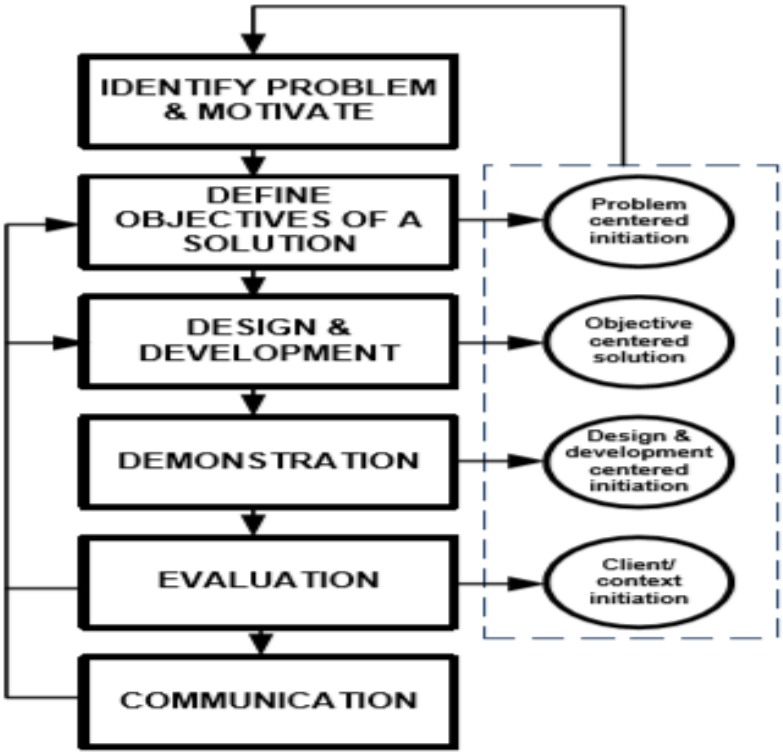


Figure 3.1 A Design Science Research Process Model (Peffers et. al, 2007)

The modification of the original model presented by Peffers et. al, (2007) was inspired from the fact that there is enough literature suggesting the importance of software process improvement for small firms. It is evidence in literature that there are a lot of software process improvement frameworks (Pressman, 2009). Hence the research does not seek to unearth the importance of the framework in context but rather suggests a new approach (using CMMI V2.0 as a model for process improvement and DevOps as a software development method) to develop a framework using existing outmoded frameworks to create a more enhanced and modern artifact that is in existence and well known.

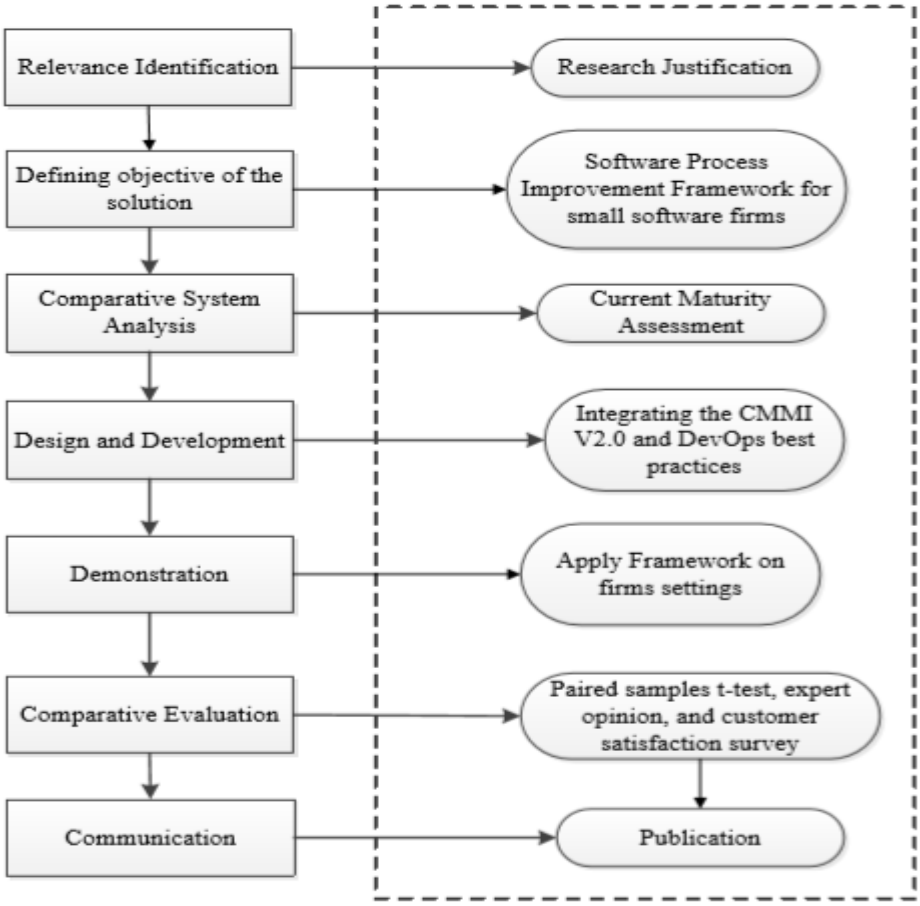


Figure 3.2 Proposed Design Science Research Methodology

The methodology shown in figure 3.2 was proposed as a design science research methodology followed throughout this study that includes the relevance identification stage that represents the process of establishing the need to improve an existing software process to enhance its quality of software products. The comparative system analysis is to compare and establish clearly the missing practice areas of inside small firms verses the well-known state-of-the-art or best recommended practices guided by CMMI V2.0 and DevOps. Once the required practices are known, the software

process improvement framework is designed and developed. A comparative evaluation of the software process improvement framework is done with the requirements of the well-known one using paired samples t-test by taking the maturity levels of each firms before and after the implementation of the framework and the results of the evaluation as well as the missing software practices are communicated as the final stage of this study's design science research methodology.

3.3 Problem Identification and Motivation

This encompasses defining the specific research problem and justifying the value of a solution. Since the problem definition is used to develop an effective artifactual solution, it may be useful to atomize the problem conceptually so that the solution can capture the problem's complexity. Justifying the value of a solution accomplishes two things: it motivates the researcher and the audience of the research to pursue the solution and to accept the results and it helps to understand the reasoning associated with the researcher's understanding of the problem. Resources required for this activity include knowledge of the state of the problem and the importance of its solution.

According to literatures and targeted interviews, small firms found hard to define, enact, formalize since it requires specializing skills about software processes, mastering special notations and tools (Johnson, 1998, Dangle et al., 2005, Khan et al 2010) and institutionalize the frameworks since it needs to match the culture of small-scale software development (i.e., with short deadlines, dynamic projects, and tight budgets) and their settings (Oktaba et al., 2007; Scott et al., 2001). CMMI only focuses on what to do at an organizational level, which lacks the descriptions on how software products are developed. It only focuses on what to do for the improvement rather that how to do for the improvement (Niazi et al, 2003). Small-scale firms need external assistance to adopt and implement standards (Laporte et. al., 2008a), because small software firms have different characteristics compared to other sizes of software firms such as number of employees, capital, recourses, tools used, methods adopted (Hofer, 2002).

3.4 Defining objectives of a solution

The objective was to develop a software process improvement solution for the improvement of the way software practitioners develop products that enables them to produce a better quality product and provide software products for users which cultivate and addressed their real needs. The software process improvement framework provides a mechanism for assessing the firms' current maturity level with respect to CMMI V2.0 and DevOps categories. It also recommends best software practices for firms according to their assessment result on practices that shows a gap to be supported. After

the installation of the recommended best practices into the firm setting, a validation mechanism is also provided to check whether a program results in a significant improvement.

So as to achieve the above stated objective, data were gathered from three small firms. Data were collected from a variety of sources using questionnaire and semi- structured interview. The source for data was Ethiopian software firms with less than 20 employees, supplier of software for local or international market, and that have in-house development of software. From different data collection method, questionnaire survey was used because of its lower cost and time. Questions were prepared using the key process areas considerations in the four CMMI version 2.0 categories; doing, managing, enabling, and improving, and DevOps Focus Areas Key Process Characteristics which includes culture, automation, measurement, and sharing.

The questionnaire was sent to 3 software firms after getting permission from these firms, and expected to respond over one week. Those firms were selected based upon their willingness to apply the intended framework on their software development phase and willingness to implement the recommendations of the SPI framework and give data during assessment and validation stages. Only those firms were selected from the available software firms which suit the best for the study, i.e. purposive sampling technique was used to choose these firms and stratified sampling to select respondents from three firms. The respondents were divided into separate groups, called strata. Then, a probability sample (often a simple random sample) is drawn from each group. The data were collected from those firms with three groups of practitioners. The first group was made up of designers/testers/programmers/analysts, referred to as ‘‘developers’’. The second group was made up of team leaders/project managers, referred to as ‘‘managers’’. The third group was “users” which contains the real end users and customers of the firms. After data collection, data was analyzed to improve and design new features of the processes. The following table shows the characteristics of respondents:

Table 3.1 List of respondents

Developers Division		**Managers Division**		**Users Division**	
Teams	No of Respondents	Teams	No of Respondents	Teams	No of Respondents
Analysts	1	Leaders	1	End-users	4

Designers	2	Project managers	1		
Programmers	3				
Testers	1				

Current process assessment instruments, such as the questionnaire used, are needed to support the collection and analysis of information from an assessment, maintain a record of results, and provide information for assessment postmortem analysis. Use of a questionnaire supports CMM Appraisal Framework compliance (Masters, and Bothwell, 1995), facilitates integration with other process assessment instruments (Zubrow et al., 1994), ensures assessment coverage of all attributes identified in each maturity goal for each level of the CMMI, provides a framework in which to collect and store assessment data, and provides guidelines for the assessors as to which areas should be the focus of an interview. It should be noted that the questionnaire is not the sole source of input for determining CMMI rank and generating process assessment results.

The data from completed questionnaires must be augmented and confirmed using information collected from interviews and presentations, as well as by inspection of relevant documents. The questionnaire consists of five parts: 1) general description about the study performed and an invitation for responding the questionnaire accordingly 2) respondent background; 3) development process questions, 4) Software Process Measurement Questions, and 5) Software Process Maturity Questions. Part 2 of the questionnaire is used to gather information about the respondent, the firms, and the units that were involved in the assessment. The software process maturity questions in part 5 are organized by CMMI version 2.0 levels and DevOps approach. The questions are designed to determine the extent to which the firm has mechanisms in place to achieve the maturity goals and resolve maturity issues at each maturity level.

3.5 Design and Development an Artifact

A framework, generally, provides structured mechanisms to define phenomena in the research and link how they relate to each other (Weick, 1995). A conceptual framework is a tool for explaining, either graphically or in a narrative form, the main constructs to be studied i.e. the key factors, constructs and relationships between them (Miles & Huberman, 1999). The artifact is the software process improvement framework that supports and improves the firms' software process. The researcher collects best practices from CMMI V2.0 model for process improvement and DevOps software development method to develop the framework. The design mainly consists of 4 cyclic

steps as shown in the figure 3.3, while these steps can be broken down into more steps according to the method and techniques used.

3.6 Demonstration

After developing proof-of-concept level software practices, the framework was extensively adapted to be followed by three small firms. The following case descriptions provide additional detail about the performance results achieved by 3 small firms that have used CMMI models to guide their process improvement efforts DevOps software development method. The results include examples for CMMI V2.0 and DevOps maturity level of the firms and all performance categories. Each description provides background information about the participating firms and its products and prior experience with process improvement. The background information is followed by additional detail about their firm's implementation of practices advocated by the CMMI model and DevOps. The developed framework has been implemented in three small software firms hosted in Ethiopia having lower than 20 employees. These firms include Boost software development plc, Abebe and his friends' software development S.C located in Bahir Dar ICT Business Incubation Center, and Ethiopian Airlines software development team, located at Addis Ababa.

Ethiopian Airlines Software Development Team

The firm is established a software development team for more than 15 years and is associated in developing different software for various serving quality service for the clients with due credibility. In spite of long experience and expertise, it has not adapted CMMI or SPICE assessment to know the organizational capabilities. It only follows the standard process for development as mentioned in the guidelines. It was also ascertained that, the authorities have good understanding on the various process improvisation tools but they are reluctant to adapt either of the tool, as the evaluation through the international standard may lose the credibility of the organization status, which may lead to client loss. However, it was open to implement the proposed SPI framework, which is largely adapted from CMMI model. In this firm 3 respondents were selected for an assessment based on their working experience that they work for more than 2 years in the firm out of the total 20 team members. The project type selected for assessment is web based application. The project takes 3-4 months.

Boost Software Development plc.

It is engaged in developing independent software products for multiple industrial domains, mainly aimed at providing high-quality software for clients around Bahir Dar which focused on developing

a web-based application for different organizations by requests like hotels, colleges, governmental offices and personal owned businesses. They mainly used PHP, JS, and Python as their programming languages for developing applications. In this firm, employees said that they didn't receive a process improvement training yet and had no any experience on DevOps approach. However, they have applied a bootstrap framework for the improvement of their software process. In this firm 2 respondents were selected for an assessment based on their working experience that they work for more than 3 years in the firm out of 9 employees. The project type selected for assessment is web based application. The project takes 3-4 months.

Abebe and His friends' software development s.c

A shared company consisting of 6 employees and established with the purpose of providing Information system development largely focused on web application development using tools and languages like angular, MongoDB, NodeJS, JavaScript, and python, systems integration, and product implementation. It develops custom-tailored applications for the automotive, medical, pharmaceutical, real estate, property management, banking and finance, and retail industries. Product implementation services include analysis of user environments, followed by custom tailoring of its products. Other services include providing quality assurance, network support and training, and writing technical and user documentation. They had received process improvement training as a course in related program of study. In this firm 2 respondents were selected for an assessment based on their working experience that they work for more than 4 years in the firm. The project type selected for assessment is web based application. The project takes 3-4 months.

3.7 Evaluation

The study involves developing the framework and evaluating its performance. The framework was well tested and evaluated by to help firms improve their software development processes. SPI process evaluation assesses the degree to which changes have been instantiated and adopted, the degree to which such changes result in better software quality or other tangible process benefits, and the overall status of the process and the organizational culture as SPI activities proceed (Sommerville, 2011). An assessment is a diagnostic tool to aid organizational improvement. Its objectives are to provide a clear and factual understanding of the organization's state of software practice, to identify key areas for improvement, and to initiate actions to make these improvements. The assessment starts with the senior manager's commitment to support software process improvement (Humphrey, 1993). After making the assessment and analyzing the measurements, firms can start improving their processes. Primarily, the research was validated by obtaining

empirical data from local software firms, and this data were further being validated by conducting interview.

The validation process involved case study work on three small firms identified before. The case studies consisted of 6 monthly visits. The initial visit to the firms was to walk through the process and identify the problems that could be reduced through the development of the framework. From discussions with the people involved in the case study work, it was soon evident that there were problems within the software process in their firms. Then the framework was designed so as to reduce the aforementioned and identified problems during the first visit. The outcome of SPI framework developed was evaluated by comparing the success indicators' values with their corresponding metrics before and after the SPI framework applied in those firms. I.e. Pre-Post Comparison evaluation method is used (Birisci et al., 2009; Sommerville and Ransom, 2005) by applying the framework in live projects as dynamic validation techniques. In the context of experimental design, Wohlin et al, (2000) defined confounding factors as variables that may affect the dependent variables without the knowledge of the researcher. They represent a threat to the internal validity of the experiment and to the causal inferences that could be drawn since the effect of the treatment cannot be attributed solely to the independent variable. As shown in Figure 3.3, both independent variables (treatments) and confounding factors represent the input to the experiment and the assessment validity of the dependent variables (effects) is threatened (Pearl, 1998).

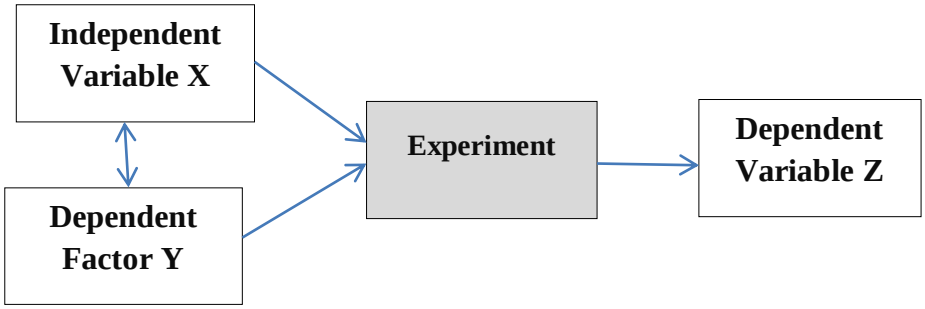


Figure 3.3 The Influence of Cofounding factors (Pearl, 1998)

Assuming that in the evaluation of software process improvements the change is assessed by comparing indicators which represent an attribute before and after the initiative has taken place, it is apparent that the problem of confounding factors, as it is encountered in an experimental setting, is also an issue in the evaluation of SPI initiatives. It is argue therefore that it is of paramount importance to identify potential confounding factors in the field of software process improvement. Kitchenham et al. (1995) identified several confounding factors in the context of the evaluation of software engineering methods and tools through case studies.

Table 3.2 Cofounding factors and remedies (Kitchenham et al. 1995)

Cofounding factor	Description	Remedy
Learning bias	The initial effort to acquire the knowledge for using the method or tool interferes with the evaluation of its benefits.	Separate activities aimed at learning the new technology from those aimed at evaluating it.
Participant bias	Attitude of the case study participants towards the new technology (enthusiasm Vs. skepticism).	Select participants according to a standard staff allocation method.
Project bias	The projects on which the technology is evaluated differ on application domain, e.g. embedded-systems and information systems.	Select projects within the same application domain and the same project size.

Altogether there would be 4 Process Dimensions under CMMI V2.0, in each process dimension, there are 20 practice areas and each area is further characterized by descriptive questions relating to the processes are being covered in respective phases. Each question is awarded with score range from 0 to 5, based on the response. The process was rated as per the attainment such as 0-Not attained at all, 1- poorly attained, 2-fairly attained, 3- attained averagely, 4-Largly attained, 5- completely attained The total score obtained in each process dimension, determines the capability of the firms. According to Sharmistha et al. (2012), further the scoring table has been grouped in five maturity levels i.e. optimized, quantitatively managed, defined, managed and initial, based on the score obtained. The score from 0 to 50% initial, 51% to 65% managed, 66% to 80% rated defined, 81% to 90% quantitatively managed and above 90% rated optimized. This scoring chat would enable to establish their maturity status.

The outcome of SPI initiative is evaluated by comparing the success indicators' values before and after the SPI initiatives took place. The means of data from two related samples; say, observations before and after an intervention on the same participant; comparison of measurements from the same participant were compared. The assessment result for firms before (pre) and after (post) the SPI initiative intervention was recorded. Results are continuous (scale) data are often summarized by

giving their average and standard deviation (SD) using SPSS software platform, and the paired sample t-test is used to compare the means of the two samples of related data. The paired sample t-test compares the mean difference of the values to zero. It depends on the mean difference, the variability of the differences and the number of data. The paired sample t-test, sometimes called the dependent sample t-test, is a statistical procedure used to determine whether the mean difference between two sets of observations is zero. In a paired sample t-test, each subject or entity is measured twice, resulting in pairs of observations. The performance of a sample of respondents before and after completing the SPI initiative was measured, and the differences were analyzed using a paired sample t-test (Statistics Solutions, 2019).

$$t = \frac{\bar{d}}{s_d / \sqrt{n}}$$ Where,

$$\bar{d} = \frac{\sum d_i}{n}$$

$$s_d = \sqrt{\frac{\sum d_i^2 - n\bar{d}^2}{n-1}}$$

d_i is case-wise difference

n is sample size

$\overline{d}$ is the mean of the case-wise differences

Sd is standard deviation of case-wise differences

t is the test statistic for the paired t-test.

Secondly, the research work was validated by presenting the framework to interested software experts, and feedback was communicated back to the researcher for further modification of the framework. Research recommendations were conveyed to all the interviewees, and some experienced developers, process engineer and quality assurance analysts were asked if the recommended framework were practicable; i.e. static validation evaluation method was used which was done through a presentation of the framework to experts whose feedback is gathered regarding its applicability in real projects inside the firms selected for investigation. The SPI framework should

be validated to ensure its credibility. This is done by means of a validation method called expert opinion in which a framework is subjected to experts, "who imagine how such the framework interacted with the problem contexts imagined by them and then predicted what effects they think this would have. If the predicted effects do not satisfy requirements, this is a reason to redesign the framework" (Wieringa, 2014). At last the framework was also validated by collecting customer feedbacks to judge their satisfaction level using a platform developed for this purpose.

To ensure framework accuracy, validation of the framework inputs and outputs are required. In this part, the applicability of the developed framework components for small scale software developing firms is under evaluation by using expert opinion and pre-post comparison evaluation methods (Birisci et al., 2009). A method using expert judgment has been practiced for the evaluation of the software process improvement framework developed. The underlying principle of the approach is the encoding of experts' tacit knowledge into probabilistic measures associated with the achievement level of software quality attributes. This suggests that the evaluation of software process improvement framework may be based on expert group decision-making rather than empirical evidence alone. A software expert is capable of expressing his/her opinion on the achievement level of a quality attribute based on the mental model of the software process improvement framework developed. In this study both the in-house experts and outsider experts for both the three firms were contacted. The feedbacks given by those experts were used as an input for the modification and refinement of the developed framework. Their feedbacks collected and documented as an expert opinion report and used for providing expert opinion in a field of debate, indicating points to consider in a decision process, and compiling the existing knowledge relevant for the answer to a question or the solution to a problem.

A list of common gaps or reservations regarding the establishment of the SPI framework, and a list of recommendations for improvement were mainly the results of the expert opinion sessions. The framework evaluated and any background material on what the framework aims to do, and who its target users was gathered and prepared for presentation to the experts. During demonstration the experts were instructed about the purpose and intended use of the framework. User group and task characteristics should always have presented as background information for the demonstration. During and after the demonstration the experts have recommended some improvements to be made on the SPI framework. The raw results of the evaluation were the list of detected problems. In a feedback received from the end users working in these firms, they qualified themselves as very

satisfied with the framework components, although they pointed out that the SPI framework still has room for improvement regarding its usability.

3.8 Communication

According to Nguyen (2020) the usefulness of scientific knowledge is limited if that knowledge is not communicated to other people. This research result will be submitted to the faculty as partial fulfillment of MSc degree in Software Engineering and defended. The contributions of this effort were disseminated in peer reviewed scholarly publications which only publish articles that pass a certain standard of quality and waiting to be published and to present the results of the SPI framework at university weekly seminars, national and international conferences like ENASE 2020 where other scientists can listen to presentations. The findings and recommendations of the developed framework is also placed in the firms' catalog so that they may use the best practices those recommended by the study for their future use. Records of the framework were kept on the computer, so that anyone can see them. The software process improvement framework is on the way to be published in academic journals since it raised interesting concepts in software development domain. Further, the framework was used to assess the small software development firms in a subsequent case study to continually improve their software development process.

CHAPTER FOUR

PROBLEM IDENTIFICATION AND SOLUTION OBJECTIVE DEFINITION

4.1 Problems Identification

This chapter describes the major problems identified and the objectives that the solution achieved so that the problems are solved. According to the interview, questionnaire and literatures review, the following are the major problems in small scale software firms towards improving their software development process and the solution objectives achieved.

Whiles many large organizations are enjoying the benefits of software process improvement frameworks (Turner Consulting Group, 2007), small scale software firms, specifically those in developing countries, are still faces difficulty towards process improvement (Kunda, & Brooks, 2000). There are many identified problems that small software firms faced towards software process improvement initiatives. Among them they lack of guidelines to approach (Laporte et. al., 2008a) and awareness of such programs and/or the benefits of these programs and interventions (Chevers, & Duggan, 2010). Small firms found the assessment of maturity level and improvement action plans for software process improvement programs often too awkward, time consuming, disruptive, and costly for small firms to implement (Niazi, & Babar, 2009; Pino et al., 2010), since the cost of CMM implementation was prohibitive for small firms.

For small firms it is also hard to define, enact, formalize since it requires specializing skills about software processes, mastering special notations and tools (Johnson, 1998, Dangle et al., 2005, Khan et al 2010) and institutionalize the frameworks since it needs to match the culture of small-scale software development (i.e., with short deadlines, dynamic projects, and tight budgets) and their settings (Oktaba et al., 2007; Scott et al., 2001). CMMI only focuses on what to do at an organizational level, which lacks the descriptions on how software products are developed. It only focuses on what to do for the improvement rather that how to do for the improvement (Niazi et al, 2003). Small-scale firms need external assistance to adopt and implement standards (Laporte et. al., 2008a), because small software firms have different characteristics compared to other sizes of software firms such as number of employees, capital, recourses, tools used, methods adopted (Hofer, 2002). The study provides an account of the investigations on how the nation's cultural, religious and linguistic diversity affects the operation of Ethiopian based small software development firms whenever they develop software products. The software development scenario in Ethiopia is at its youngest age. According to EITPA (2020) several developers have really good applications that are set to solve real issues in Ethiopia, however they are always face difficulties while placing the app to operation. How to move to the next level of making the app known and then becoming

a profitable venture by deploying the app into operation is a hard nut yet to be fractured. Full adaptation of the technology has been difficult in Ethiopia due to a restrictive information seeking culture, and diverse local languages, scripts and dialects which makes software development difficult to build user friendly interfaces in the local languages for online connectivity for promoting information sharing and developing a successful product nationwide in Ethiopia (EITPA, 2020).

4.2 Defining objectives of a solution

Having in mind the problems identified in section 4.1, the objective was developed to be achieved via the solution. i.e. a software process improvement solution for the enhancement of the way software practitioners develop products that enables them to produce a better quality product and provide software products for users in a satisfied manner which cultivate and addressed their real needs. This objective was assessed by using customer satisfaction survey (CSAT) platform developed. The software process improvement framework provides a mechanism for assessing the firms' current maturity level with respect to CMMI V2.0 and DevOps categories. So that the firm reached at an improved maturity level with respect to both CMMI V2.0 and DevOps practices. The cultural phenomena and the details of implementation of the SPI program on small firms are mainly addressed by incorporating DevOps into the firm setting. It also recommends best software practices for firms according to their assessment result on practices that shows a gap to be supported. After the installation of the recommended best practices into the firm setting, a validation mechanism is also provided to check whether a program results in a significant improvement.

In General, this chapter briefly described the problem definition framework and the solution objectives. Depending on the problems definition mentioned and the solution objectives set, the solution framework i.e. the SPI framework is designed. The following chapter details the design and development of this solution framework later evaluated according to the objectives and demonstrated to the users.

CHAPTER FIVE

FRAMEWORK DESIGN, DEVELOPMENT, DEMONSTRATION AND EVALUATION

5.1 Design and Development an Artifact

The artifact is the software process improvement framework that supports and improves the firms' software process. The researcher collects best practices from CMMI V2.0 model for process improvement and DevOps software development method to develop the framework. The design mainly consists of 4 cyclic steps as shown in the figure 5.1, while these steps can be broken down into more steps according to the method and techniques used. In most cases, however the process contains these steps.

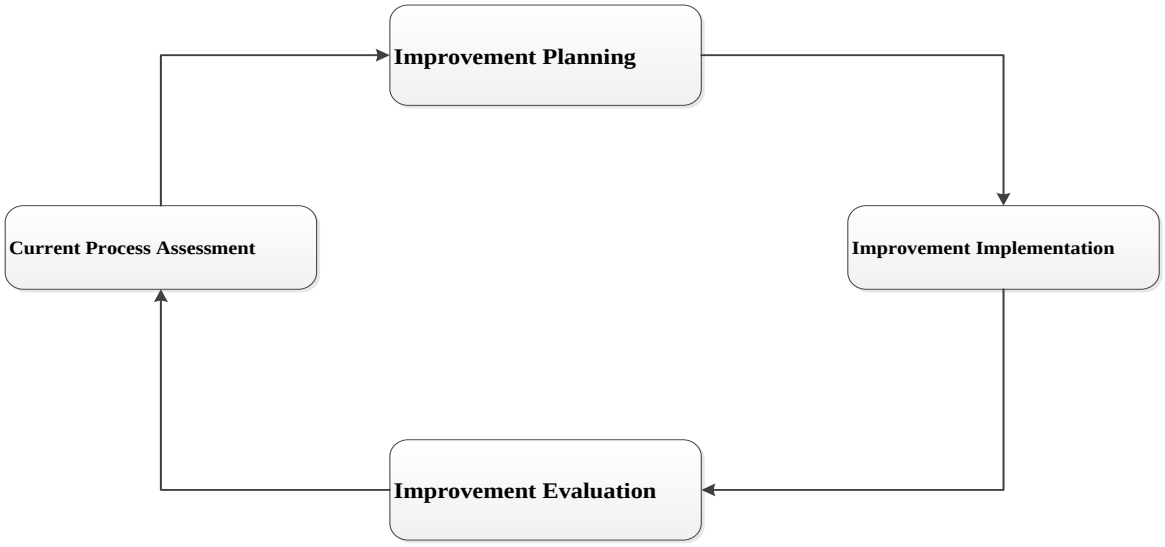


Figure 5.1 The Proposed SPI framework foundations

Current Process Assessment

After a firm is identified as a candidate for assessment and the assessing firm conducts an executive level briefing, committed for full assessment, and prepare for assessment, an assessment of the current situation of the software process was done by eliciting the requirements from the stakeholders, analyzing the existing artifacts and deliverables (quantitatively), and identifying the inefficiencies from the selected small scale software firm' process. Its objectives are to provide a clear and factual understanding of the organization's state of software practice, to identify key areas for improvement, and to initiate actions to make these improvements. The elicitation can be conducted through distributing questionnaire to the project team such as developers and project managers.

The key consideration in this step is to analyze the responses given by the project team for identifying organization goals and ask the solution-oriented questions. The measurements were identified using the pre-post comparison technique using paired samples t-test that helps in comparing the current status with the benchmark stored and measuring the effectiveness of the improvement process. Finally, the entire assessment team participates in preparing an assessment report that includes the findings and recommendations for actions to address these findings and presenting it to senior management and the assessment participants.

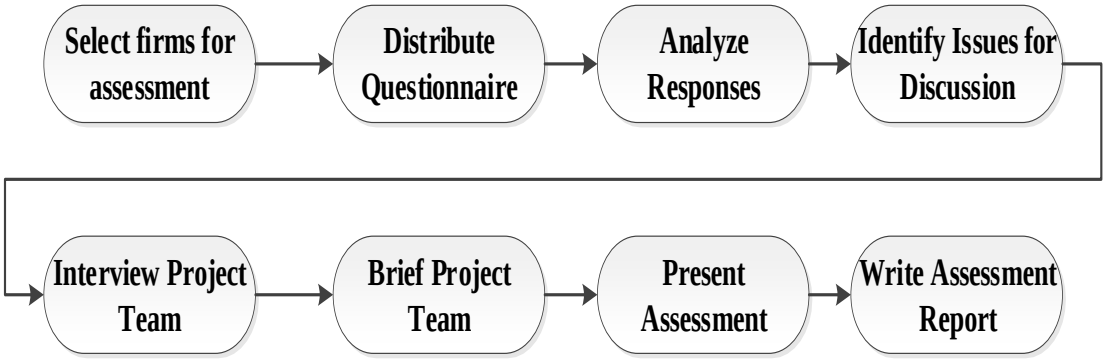


Figure 5.2 The assessment process

Improvement Planning

After the quality, cost or schedule bottlenecks have been identified and analyzing the current situation and the improvement goals, the findings should be categorized and prioritized according to which one is the most important or have the most severity. We should observe what is the new target level of improvements should look like. Moreover, in this step, the gap between the current level and the target level should be planned in terms of a set of activities to reach that target. These activities should be prioritized with the alignment of the involved stakeholders and the organization goals, for example, if the project is using the CMMI model, the target could be reaching maturity level 4 and the firm at level 3, in that case, the plan should be focused on the process areas and their activities which is related to that level of improvement with the alignment of the firm goal.

Improvement Implementation

In this step, the planned activities are executed to modify the process to remove identified bottlenecks and it puts the improvements into practice and spreads it across the firm, what can be effective at the 2nd, 3rd, and 4th step that planning and implementation could be an iterative way, for example, implementing improvement for refining requirements first, then implementing the reduction for testing process time, and so forth. This iterative way of implementation helped the firm to realize the early benefits from the SPI program early or even adopt the plan if there is no real impact measured from the improvement. In parallel

with introducing process changes in the firms, the staff involved in new process proposals should be trained about the improvement program.

Improvement Evaluation

What is cannot be measured cannot be improved, that's why in this step the impact measurement is applied compared with the target level. The evaluation considers three kind of measures, the before improvement measures, after the improvement measures, and the target improvement measure. Measurement, in general, permits a firm to compare the rate of actual change against its planned change and allocate resources based on the gaps between actual and expected progress. If the firms' process doesn't require further improvements change tuning is needed for the purpose of evolving and continuously improving process improvements.

The Proposed SPI Framework for small scale software firms is depicted in figure 5.3.

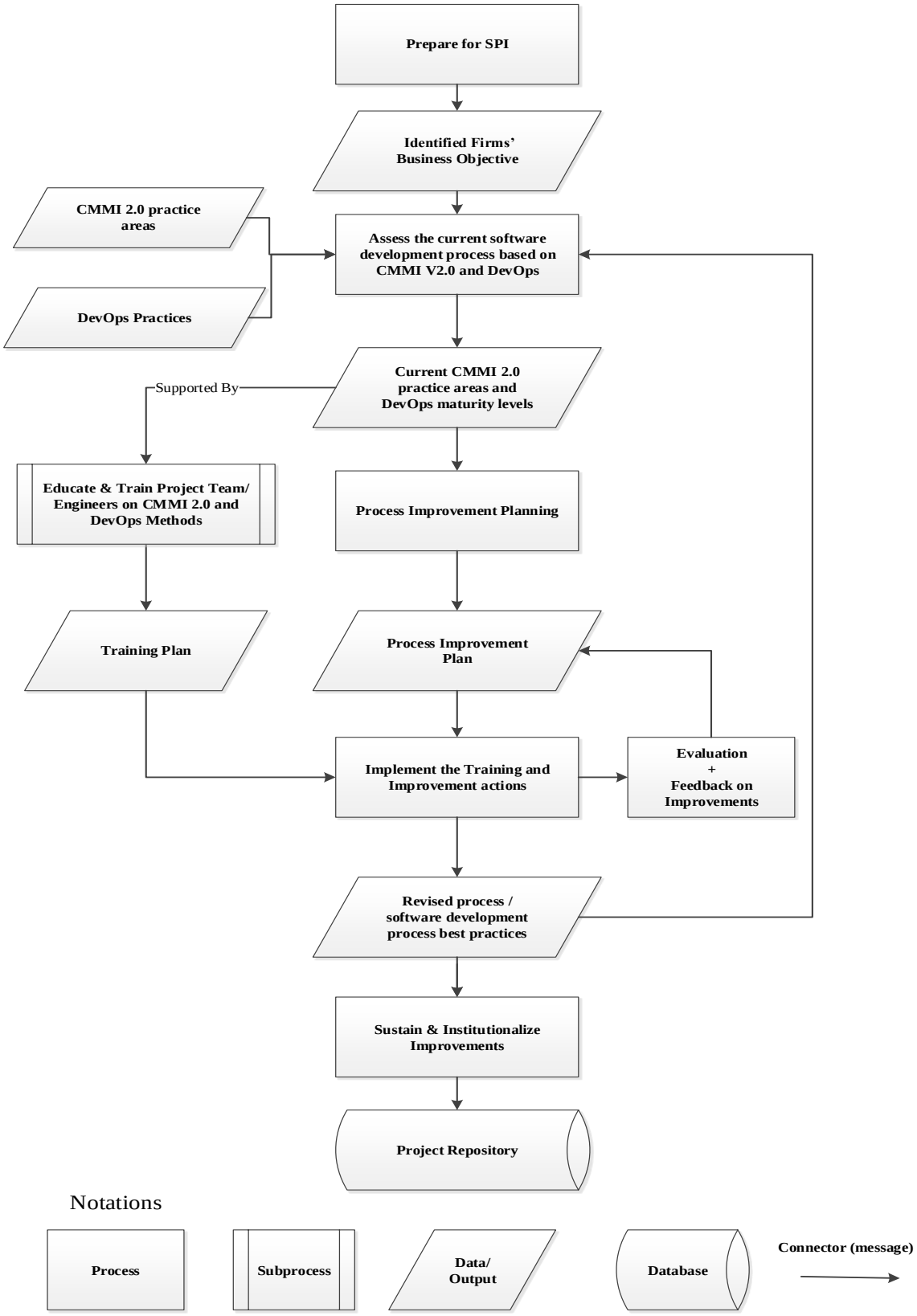


Figure 5.3 The Proposed SPI Framework

First of all, firms are expected to be committed and well prepared for implementing the software process improvement program. After their commitment towards the initiative is known and their willingness is assessed, the firms' business objective was identified so that each and every practice recommended has to be in support for their business objective and vision. Then their current process maturity was assessed so that they will enhance the low matured process from CMMI V2.0 model for improvement and DevOps software development method's best and recommended practices. After the assessment of firms' current software process maturity level, gap analysis was done to discover the problematic practices that need to be supported and improved. The difference between the local application and best practice that represents a "gap" that offers opportunities for improvement was recorded as an improvement plan with their corresponding supporting best practices. With the help of education and training with regards to the general aim of the improvement program, generic concepts and methods to provide professionals with the intellectual tools they need to apply the software process effectively and to make rational decisions about improvements to the process, the specific technology and tools that have been adopted for local use; for example, UML was chosen for analysis and design modeling on firms, then a training plan for software engineering using UML was established, and business communication and quality-related topics to all stakeholders that help enable better communication among stakeholders and foster a greater quality focus was included under a comprehensive training plan.

The improvement plan and training plan was implemented and an action was taken. CMMI V2.0 and DevOps best practices that were in support for the gap identified were defined and installed as part of a new software process culture. Evaluation and feedbacks on the improvement were needed to assess the degree to which changes have been instantiated and adopted, the degree to which such changes result in better software quality or other tangible benefits and the overall status of process and the organizational culture as SPI activities proceed. The revised software development best practices for that results in improvement according to the CMMI V2.0 and DevOps were sustained and institutionalized and stored in the firms' project repository for continuously improve their software process for the future.

The data collected through interviews and questionnaire was organized in enabling, doing, managing and improving which are the four dimensions of CMMI V2.0 and culture, automation, measurement, and sharing, which are the four dimensions of DevOps practices. Software process assessment data analysis was done using Excel. The data gathered were analyzed through descriptive statistics. Once the questionnaires were returned, they were checked for consistency and went through a data validation process. In fact, for each questionnaire, there were two aspects that were extracted and computed. These are CMMI Maturity level of each firm, and DevOps Maturity level of each firm,

Mean and Percentage were used to calculate CMMI V2.0 and DevOps maturity level of the firms. In order to calculate the process maturity level of a firm first group key practice areas by categories of CMMI V2.0 (Doing, Enabling, Managing, Improving) and DevOps (culture, automation, measurement, sharing), then the response rank using a scale of 1-5 (from 1= Low Maturity to 5= High Maturity, 0= Not Applicable) for each key practice areas of CMMI V2.0 categories and DevOps lifecycle phases were undertaken for each respondent and calculating the mean value for each CMMI V2.0 and DevOps categories. Individual area of assessment was characterized by 39 relevant questions to access the processes followed. Scoring would be taken up for each question answered from scoring scale 1-5. The best-rated process was awarded with "Five" score, the least was with "One", "Zero" for not applicable in the firm settings. Finally, the value of the maturity level was calculated by taking the mean of individual categories for both CMMI and DevOps. Taking the mean of all respondents finally indicates the maturity level of the firms which will be expressed in terms of percentage. The firms' practice and process maturity level were assessed as per internationally recognized standards which indicates where the firm is. This guides to propose on how to improve the level of maturity to meet project goals.

5.2 Assessment of Firms' Process Maturity level

The maturity assessment has been performed for the four key process areas of CMMI categories, including doing, enabling, managing and improving that are covered by the research. The four DevOps Lifecycle Phases are also the focus of the assessment including culture, automation, measurement, and sharing. Subsequent parts provide assessment summary result and discussions.

5.2.1 Maturity level assessment of key process areas of CMMI V2.0 categories

The overall maturity of key process areas of CMMI V2.0 categories by percentage before and after the intervention for firms is 63.45 and 74.31 for firm A, 61.31 and 73.59 for firm B, and 66.13 and 73.24 for firm C respectively, which show the firms are at basic level of maturity with better insights to reach software quality standards. According to the results obtained both the firms A and B improved from managed to defined level, and firm C is at defined level which doesn't achieved a significant level improvement. The analysis calculation on how the maturity level of CMMI V2.0 categories was determined is shown in appendix 2.

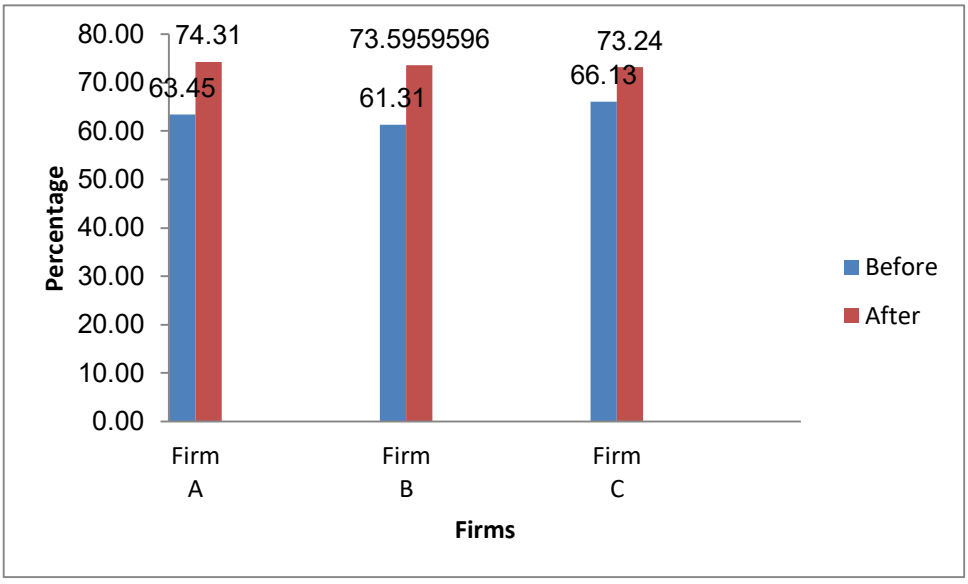


Figure 5.4 Overall CMMI V2.0 Maturity Level of firms

Maturity level of firm A

Firm A performs better at Doing, Enabling and Improving categories while remains jobs on the managing category. To reach at an improved level, the firm expected to implement the following practices and reached at defined from managed level:

- The firm should apply formal procedure for managing and planning security and secure suppliers to develop secure solutions/products.
- The firm should determine root causes of selected outcomes by following an organizational process.
- The firm needs to monitor, analyze, understand, and report on current and future demand for services, use of resources, capacity, service system performance, and service availability.
- The firm should follow a risk management plan for monitoring identified risks or opportunities and communicate status to affected stakeholders.
- Incident resolution and prevention mechanisms should be developed, keep updated, and followed.

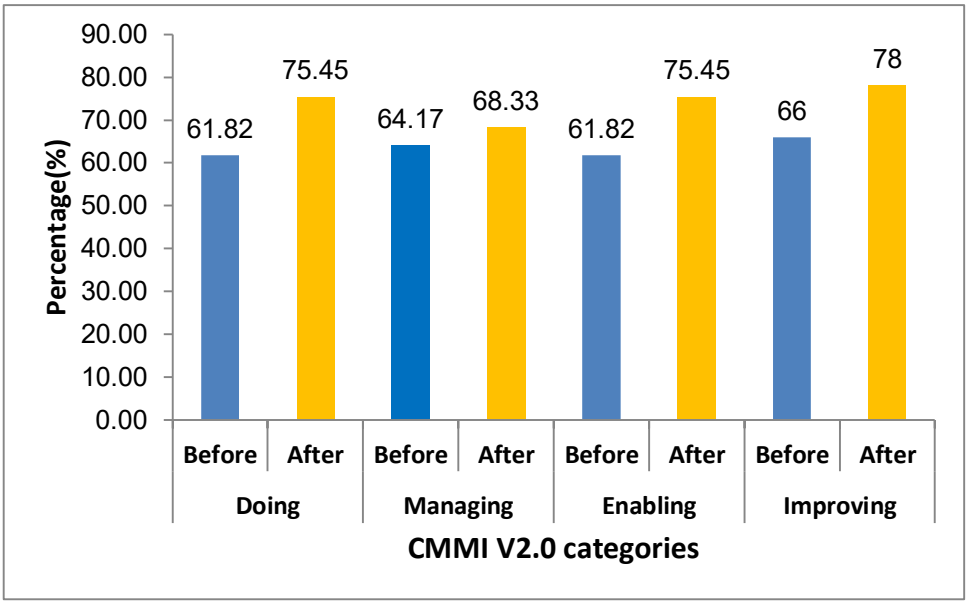


Figure 5.5 CMMI V2.0 Maturity level of firm A

Maturity level of firm B

Although it remains some gaps on enabling and improving categories, firm B performs relatively better at doing and managing categories. To reach at a level higher than the current one, it is recommended the firm to apply the following practices and achieved a defined level of maturity from the managed one:

- The firm should use quality assurance activities for software projects.
- The firm should apply a defined integration strategy and procedures to achieve complete product integration through progressive assembly of product components, in one stage or in incremental stage.
- The methods to be used in the purchasing of a product and product components should be determined.
- The senior management should identify what is important for doing the work and define the approach needed to accomplish the objectives of the organization.
- Giving a course to the people regarding statistical methods, data collection, analysis and reporting processes should be practiced.

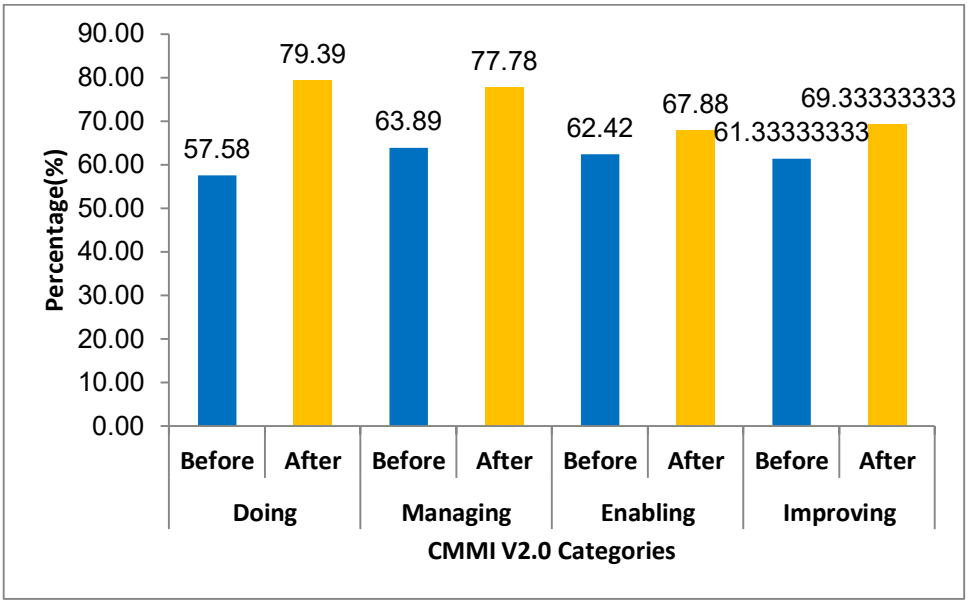


Figure 5.6 CMMI V2.0 Maturity level of firm B

Maturity level of firm C

Firm C performs an outstanding level at doing and improving, and relatively better at enabling category while remains additional jobs on the managing category. To reach at the next to level; i.e. the quantitatively managed level from its current defined level, the firm expected to implement the following practices:

- Size, effort, and cost for software projects should be determined.
- There should be a defined procedure to develop a list of tasks and assign people to tasks.
- The firm should follow a risk management plan for monitoring identified risks or opportunities and communicate status to affected stakeholders.
- The firm should invest workgroups with the responsibility and authority for determining how to conduct their business activities most effectively.
- The firm should increase the opportunities relating with providing a standard procedure for exploring and evaluating potential new processes, techniques, methods, and tools to identify improvement opportunities.

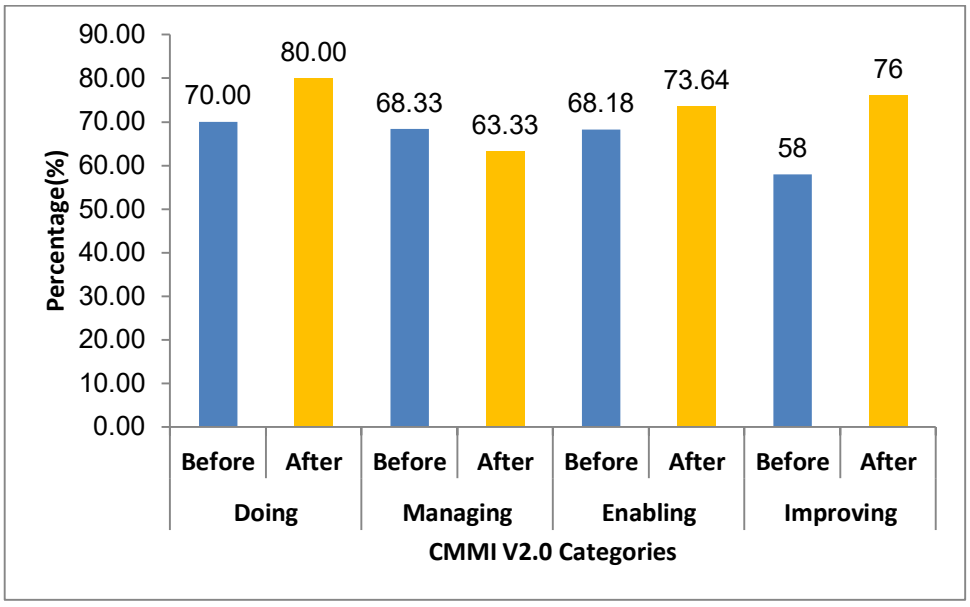


Figure 5.7 CMMI V2.0 Maturity level of firm C

5.2.2 Maturity level assessment of DevOps Lifecycle Phases

As can be seen in the figure below, the process maturity level of the firms is at basic process striving to reach the industry standards. The research finding indicates that the overall maturity of key process areas of DevOps categories by percentage before and after the intervention for firms is 65 and 77 for firm A, 72.33 and 83.33 for firm B, and 71.5 and 81 for firm C respectively. According to the results obtained firm B and C improved from defined to quantitatively managed level while firm A achieves a significant level of maturity from managed to defined level.

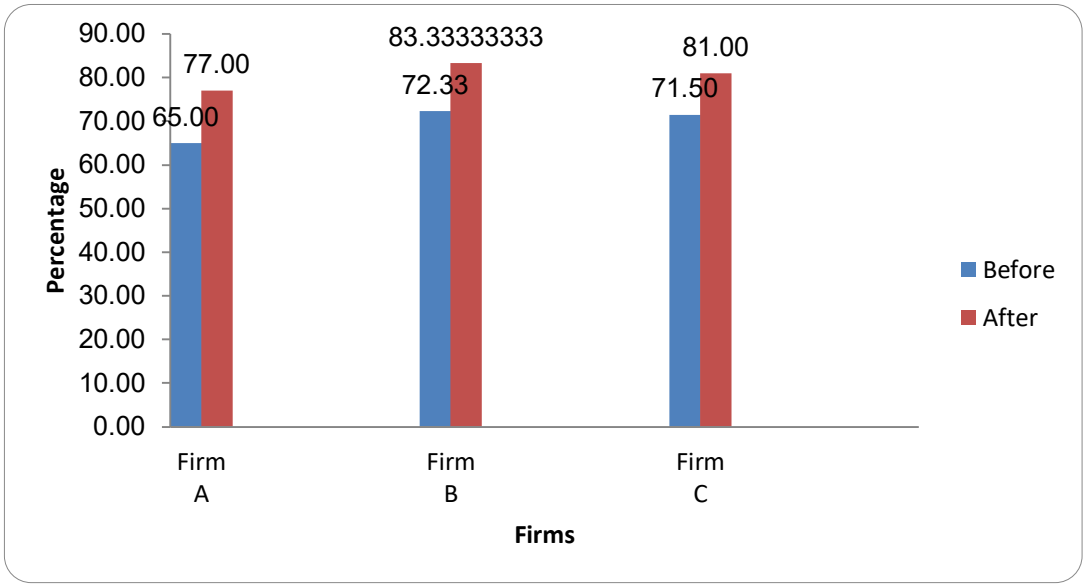


Figure 5.8 Overall DevOps Maturity of firms

DevOps Maturity of firm A

The total maturity level of firm A in percentage before the intervention is 65 %. This indicates that the firm is at managed level. The firms' maturity level for each DevOps categories culture, automation, measurement, and sharing before and after the intervention by percentage is 64, 78, 56, 62 and 74, 90, 64, 80 respectively. The firm performs better at automation and sharing while still remains jobs on its practices relating culture and measurement. Even if the study recommended the firm to insure the following practices so that it reaches to the highest level as possible, the firm reaches 77 % percent that shows there is significant improvement shown due to the introduction of these practices:

- Quality measures before deployment are not aligned with after deployment. So, try to align these measures.
- A continuously-updated dashboard should be used to show progress of feature.
- The firm should measure the time to go from development to deployment.

- The development and operations teams should collaborate during a production issue.
- All the members of the product team should have access to code, status, metrics and history of the project that is running.

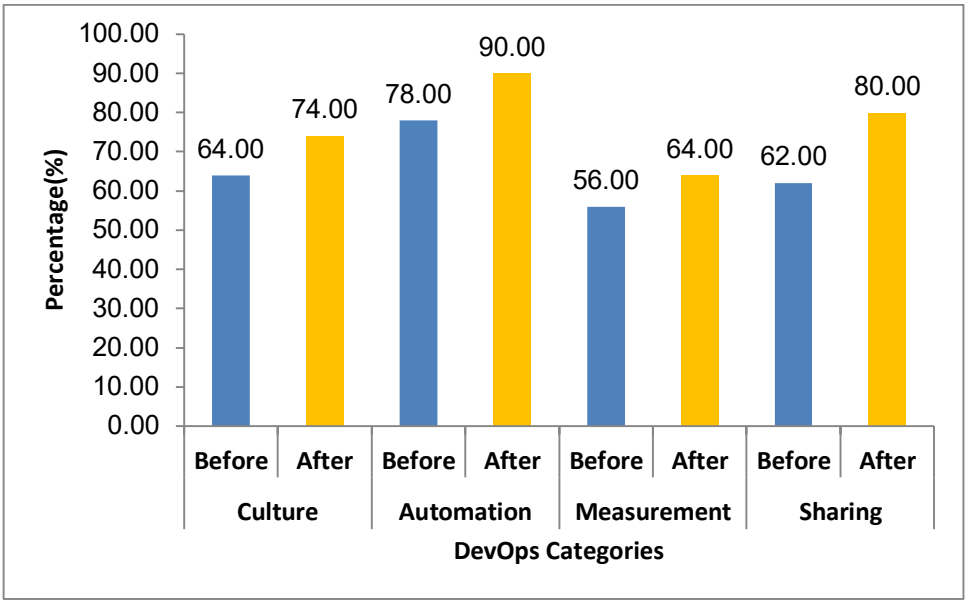


Figure 5.9 DevOps Maturity of firm A

DevOps Maturity of firm B

The total maturity level of firm B in percentage before the intervention is 72.33%. This indicates that the firm is at defined level. The firms' maturity level for each DevOps categories including culture, automation, measurement, and sharing before and after the intervention by percentage is 68, 68, 84, 69.33 and 81.33, 74.67, 90.67, 80.67 respectively. The firm performs better at measurement and sharing relatively while remains jobs on improving its practices relating culture and automation. So, the study recommended the firm to ensure the following practices so that it reaches to the highest level as possible and achieved a breakthrough improvement to quantitatively managed maturity level.

- The leaders most effectively help their teams gain autonomy in their work by establishing and communicating goals, but letting the team decide how the work will be done, removing roadblocks by keeping rules simple, allowing the team to change rules if the rules are obstacles to achieving the goals, and letting the team prioritize good outcomes for customers, even if it means bending the rules.
- So as to mature the current test automation of the firm, it should apply tools like selenium which is an automated browser that allows quality assurance teams to write scripts and test web products that helps to create quick bug reproduction scripts and create scripts to aid in automation-aided

exploratory testing. It runs in many browsers and operating systems and can be controlled by many programming languages and testing frameworks (Seleniumhq, 2019).

- The firm mostly applies manual processes in operations. Try to automate development and operations activities by supporting different tools like the idea of infrastructure as code that reproduce and change the state of environments in an automated fashion from information in version control rather than configuring infrastructure manually.
- It is better applying the leading open source automation server, Jenkins, which provides hundreds of plug-ins to support building, deploying and automating any project and can be used as a simple continuous integration server or turned into the continuous delivery hub for any project. It can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help and can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.
- The third recommendation for firm C regarding automation is approaches to apply a tool Git which is a Version Control System with a repository for source code management that enables working online and offline and designed to handle everything from small to very large projects with speed and efficiency. It is easy to learn and has a tiny footprint with lightning fast performance. It is used by many international companies like Facebook, Google, Microsoft, Twitter, LinkedIn, Netflix, Android, Eclipse, etc (Git, 2019).

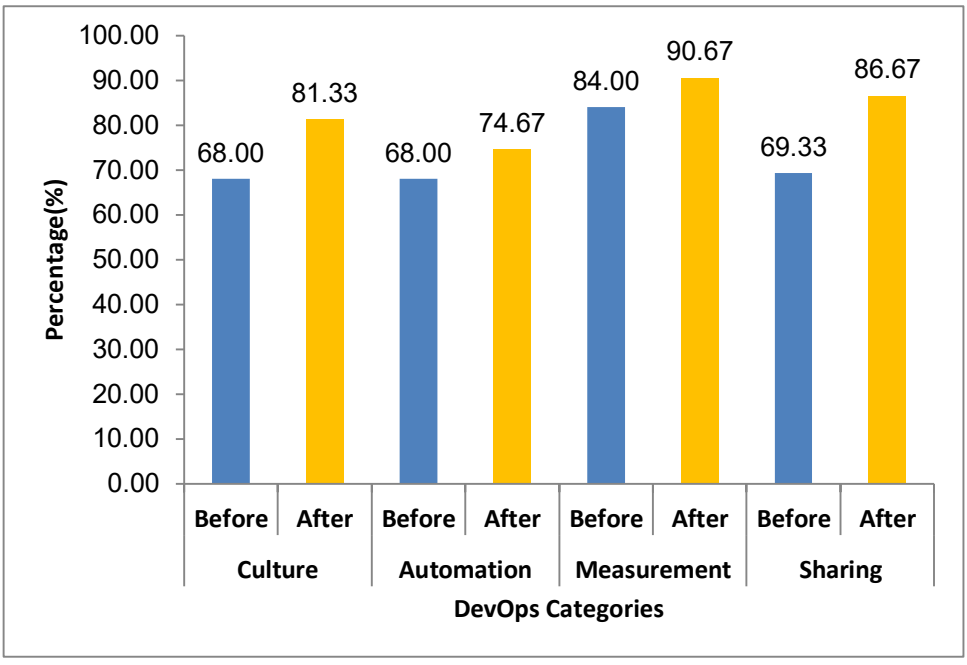


Figure 5.10 DevOps Maturity of firm B

DevOps Maturity of firm C

The total maturity level of firm C in percentage before the intervention is 71.5 %. This indicates that the firm is at defined level. The firms' maturity level for each DevOps categories culture, automation, measurement, and sharing before and after the intervention by percentage is 70, 82, 66, 68and 78, 86, 82, 78 respectively. The firm performs better at measurement and automation relatively while remains jobs on improving its practices relating to culture and sharing.

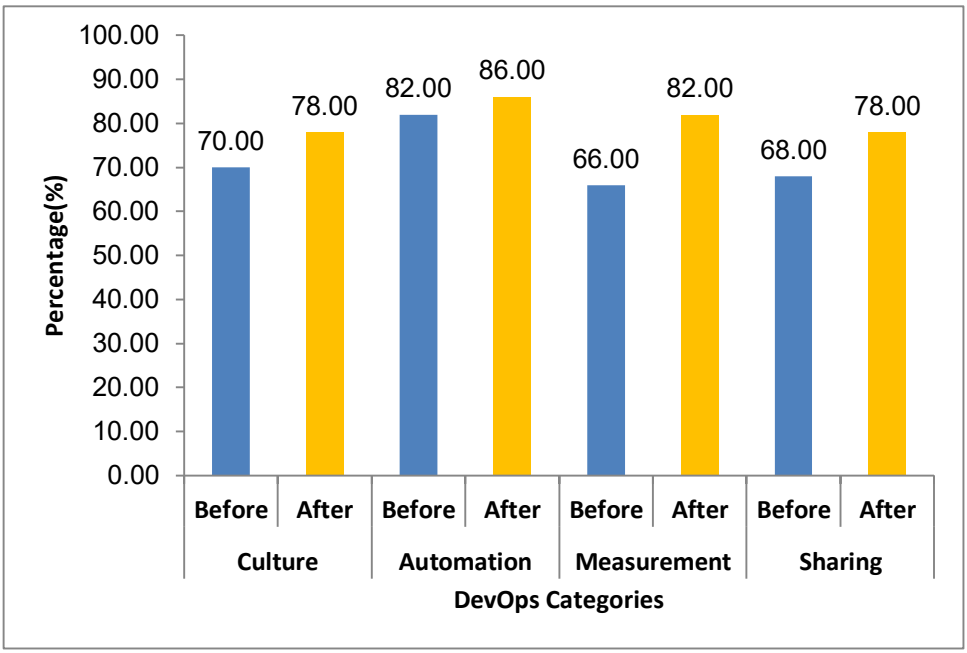


Figure 5.11 DevOps Maturity of firm C

The following best software practices were suggested so that the firm reached to the highest level as possible and achieved a breakthrough improvement to quantitatively managed maturity level.

- Culture is influenced by many factors. Among the role of leaders are the main factors that affect the way that the project team works. When leaders give their teams autonomy in their work it leads to feelings of trust and voice. Trust reflects how much a person believes their leader or manager is honest, has good motives and intentions, and treats them fairly. Voice is how strongly someone feels about their ability and their team's ability to speak up, especially during conflict. For example, when team members disagree, when there are system failures or risks, and when suggesting ideas to improve work. Trust and voice, in turn, positively affect organizational culture.
- Code, status, metrics and history should be accessible to all the members of the product team.
- As much as possible make the potential changes visible to all members of your product team.

5.3 Framework Evaluation

According to the pre-post comparison evaluation method, the value of the software process measures and the firm's process maturity levels were compared using two parameters, the pre-initiative value, and the value obtained after the framework was implemented, i.e. the post-initiative value. All participating firms carried out their project to implement CMMI V2.0 practice areas and DevOps principles and practices. Each project assumed to be lasted approximately 4 months, during which each firm was coached by a researcher. A paired-samples t-test is used when we have only one group of people (or companies, or machines etc.) and we want collect data from them on two different occasions or under two different conditions. We should assess each person on some continuous measure at Time 1, and then again at Time 2, after exposing them to some intervention (Statistics Solutions, 2019).

CMMI V2.0 Maturity Level

Firm A

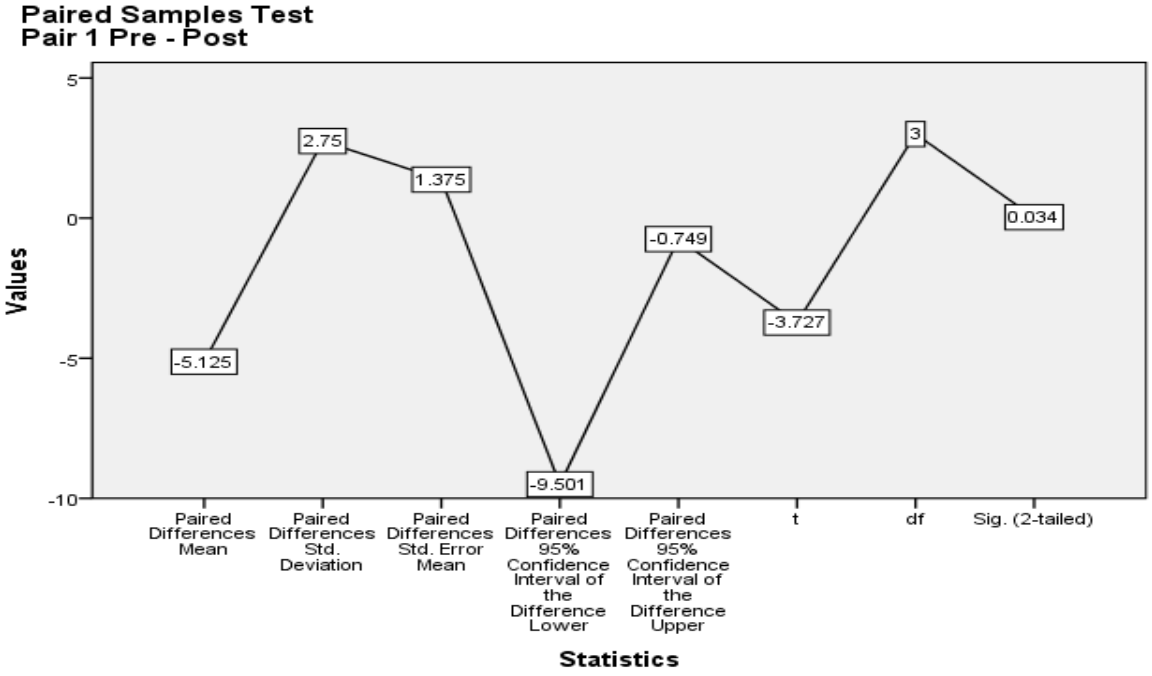


Figure 5.12 CMMI V2.0 paired-samples t-test result for firm A

Since sig-t(-3.727) i.e 0.034<α(0.05)

Therefore, it supports the objectives of the study. There is sufficcient evidence (t = -3.727, p = 0.034) to conclude that there is significant improvement in the software process of the firm. Thus the initiative is significantly effective to improve firms' software process at 0.05 level of significance.

Firm B

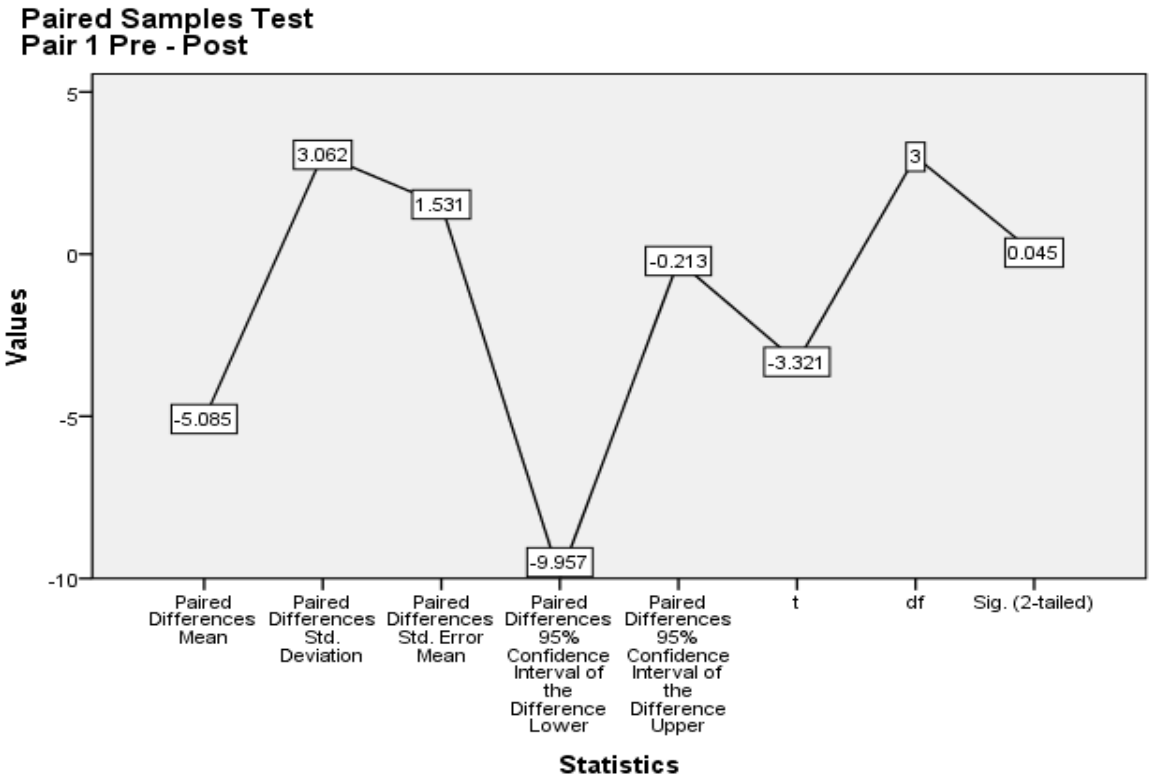


Figure 5.13 CMMI V2.0 paired-samples t-test result for firm B

Since $0.045<\alpha(.05)$

Therefore, it supports the objectives of the study.

There is sufficcient evidence ($t = -3.321$, $p = 0.045$) to conclude that there is significant improvement in the software process of the firm. Thus the initiative is significantly effective to improve firms' software process at 0.05 level of significance.

Firm C

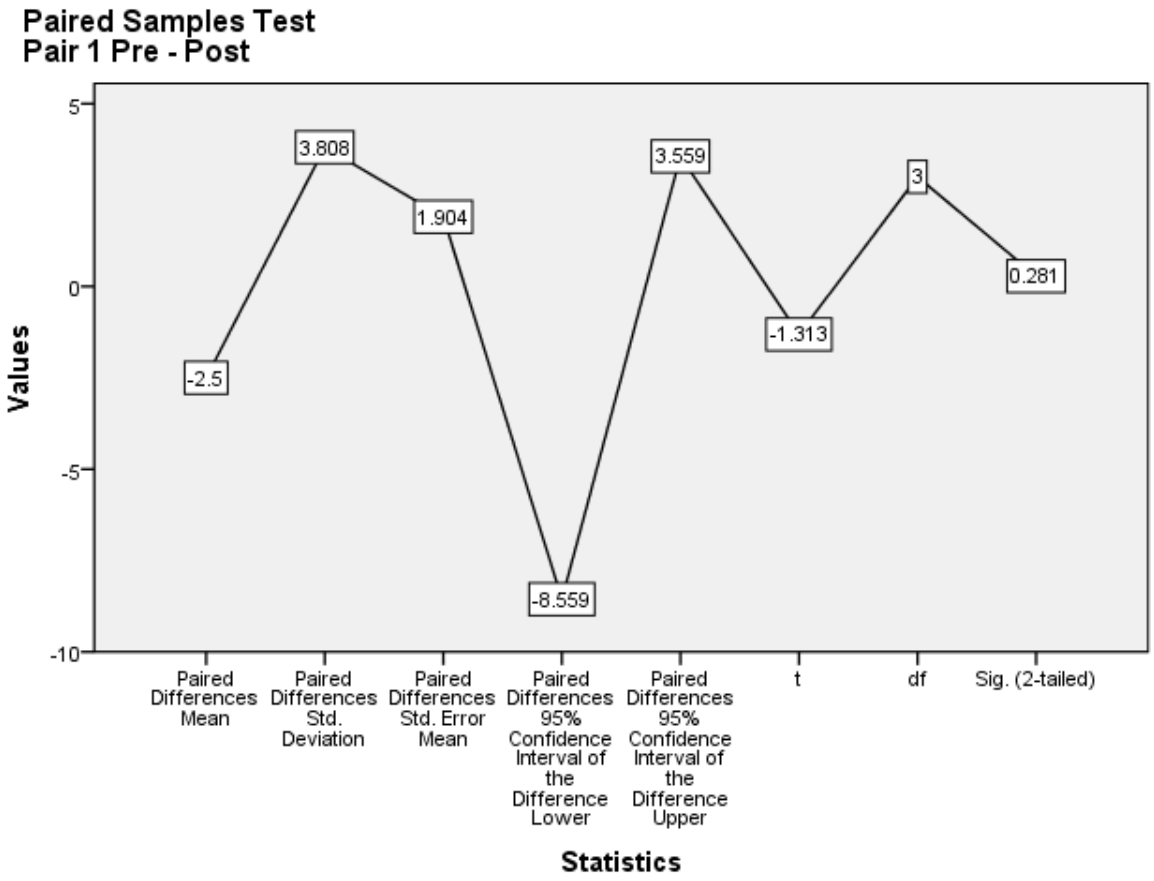


Figure 5.14 CMMI V2.0 paired-samples t-test result for firm C

Since 0.281>α(0.05)

Therefore, it argue against the objectives of the study.

There is sufficcient evidence (t = -1.313, p = 0.281) to conclude that there is no significant improvement in the software process of the firm. Thus the initiative is ineffective to improve firms' software process at 0.05 level of significance.

DevOps Maturity Level

Firm A

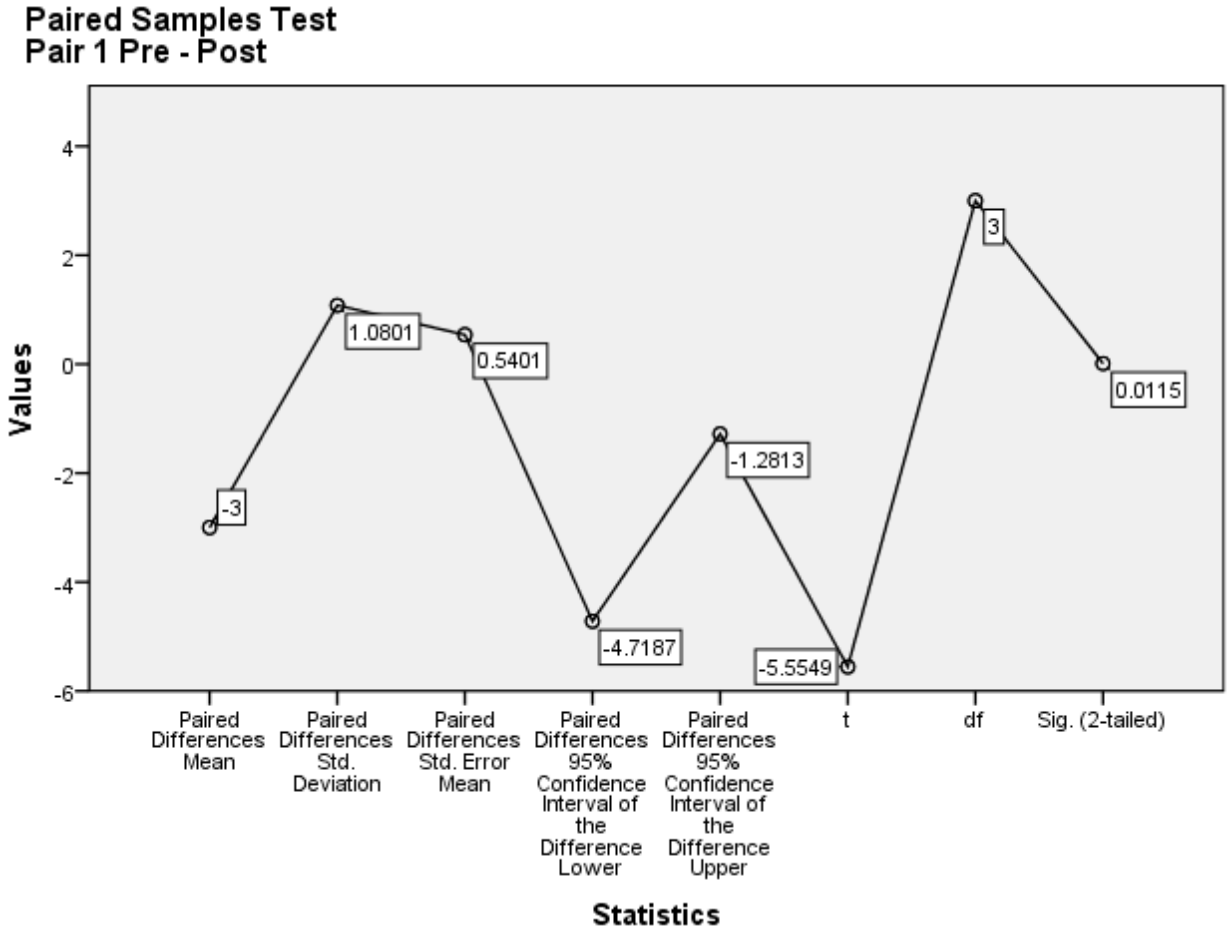


Figure 5.15 DevOps paired-samples t-test result for firm A

Since 0.012<α(.05)

Therefore, it supports the objectives of the study.

There is sufficcient evidence (t = -3.516, p = 0.012) to conclude that there is significant improvement in the software process of the firm. Thus the initiative is significantly effective to improve firms' software process at 0.05 level of significance.

Firm B

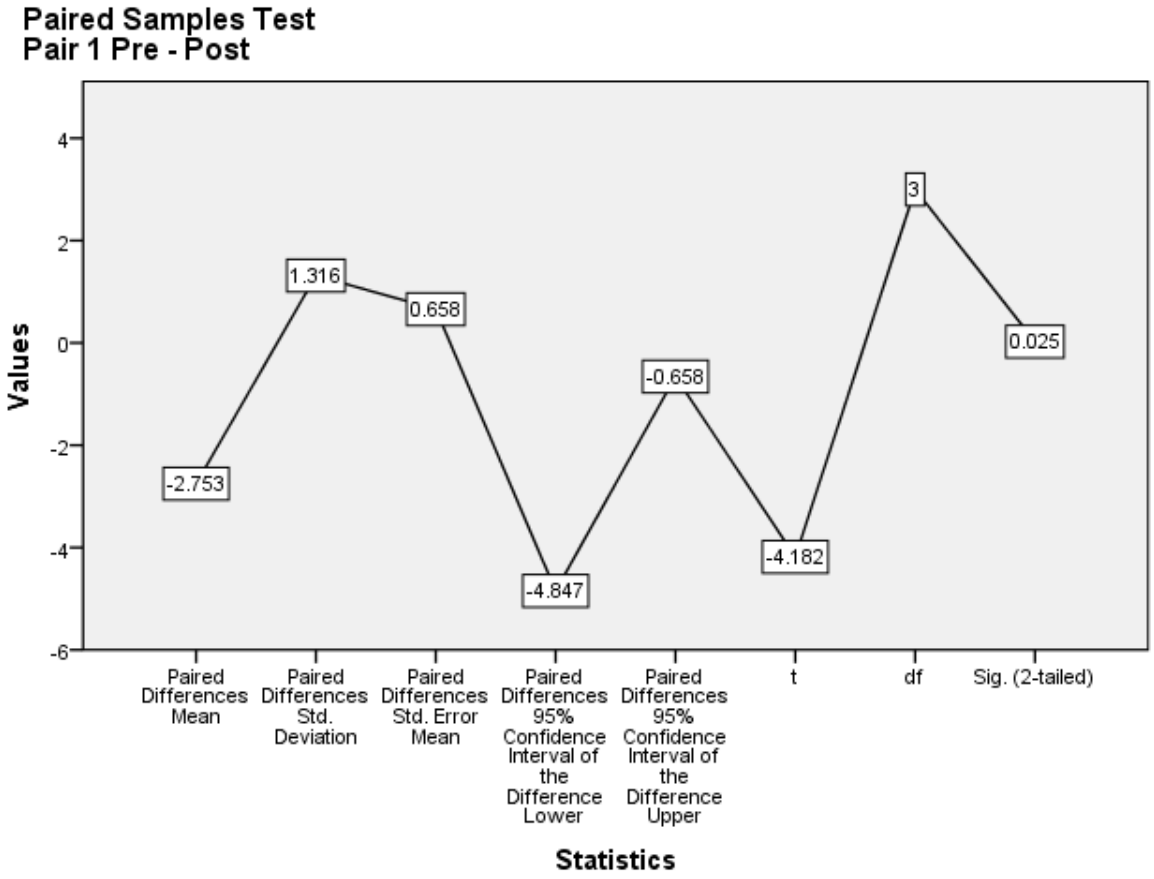


Figure 5.16 DevOps paired-samples t-test result for firm B

Since 0.025<α(.05)

Therefore, it supports the objectives of the study.

There is sufficcient evidence (t = -4.18, p = 0.025) to conclude that there is significant improvement in the software process of the firm. Thus the initiative is significantly effective to improve firms' software process at 0.05 level of significance.

Firm C

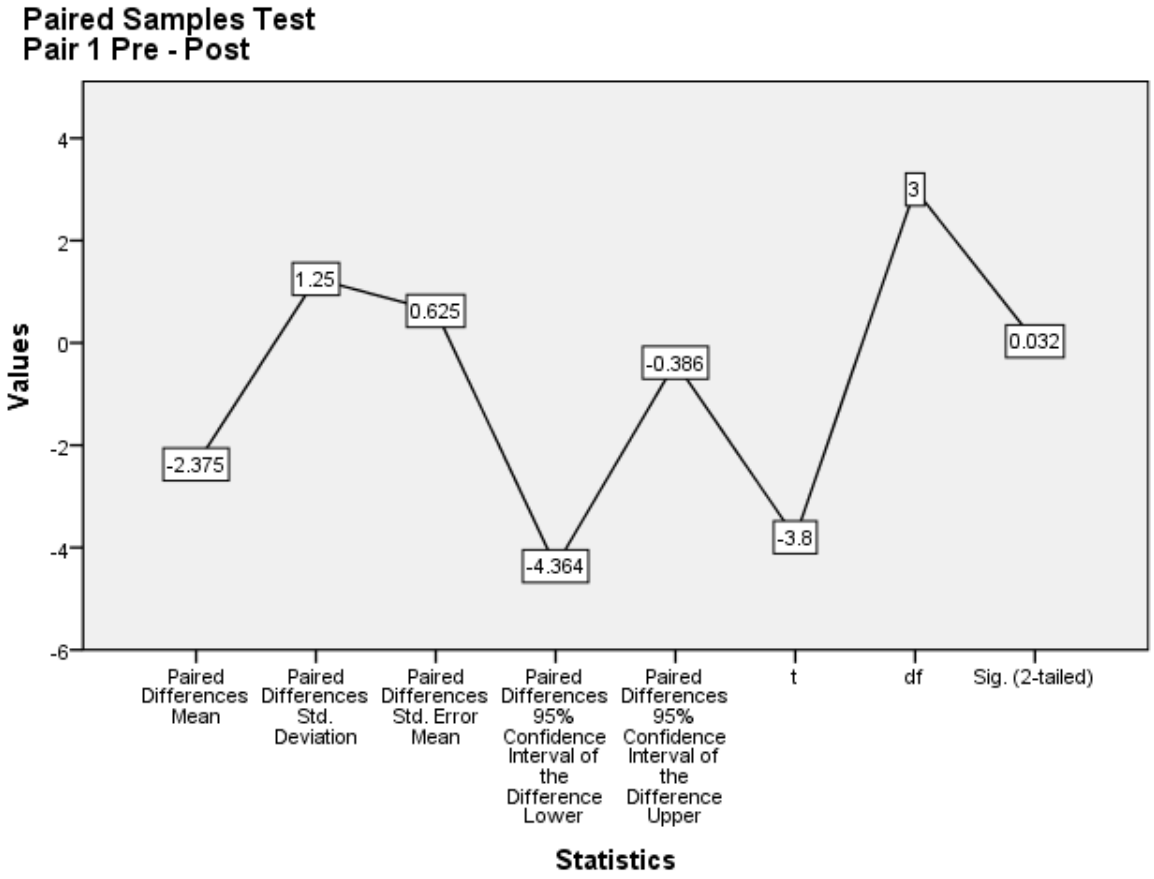


Figure 5.17 DevOps paired-samples t-test result for firm C

Since 0.032<α(.05)

Therefore, it supports the objectives of the study.

There is sufficcient evidence (t = -3.8, p = 0.032) to conclude that there is significant improvement in the software process of the firm. Thus the initiative is significantly effective to improve firms' software process at 0.05 level of significance.

5.3.1 User-Centered framework evaluation method

This category includes a method that involves users. Customer satisfaction (CSAT) platform was developed to understand firms' customer's satisfaction levels with their products, services, or experiences. According to Qualtrics (2019), this is one type of customer experience survey and can be used to gauge customers' needs, understand problems with your products and/or services, or segment customers by their score. They often use rating scales to measure changes over time, and gain a deeper understanding of whether or not we're meeting the customer's expectations. Since we have dealing with an idea or construct that ranges from zero to positive – think effectiveness – (these are known as unipolar constructs) then we have gone with a 1-5 point scale (Krosinick and Fabrigar, 1997). The response label options for this kind of question look like this:

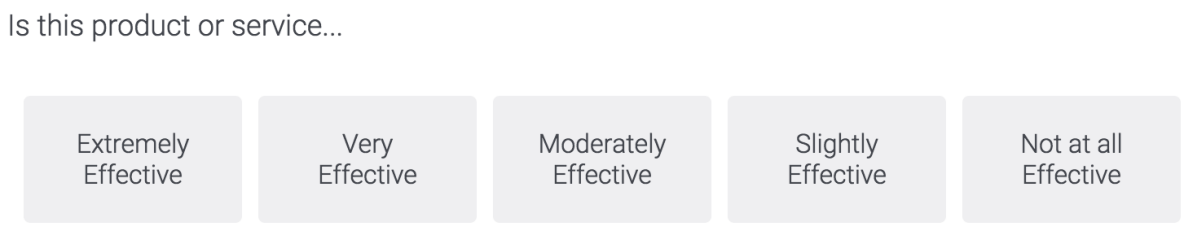


Figure 5.18 The response label options of the CSAT platform (Krosinick and Fabrigar, 1997)

The above label can be equivalently used with process maturity levels by taking "Extremely Effective" as "Optimized", "Very Effective" as "Quantitatively Managed", "Moderately Effective" as "Defined", "Slightly Effective" as "Managed", and "Not at all Effective" as "Initial" level of process maturity. This method aims to record users' performance due to the development of the SPI framework and/or users' satisfaction with, or preferences on, a framework, such as the features and the presentation of the SPI framework. Usability is defined as how easy it is to use or learn software. How easy to perform intended task, how much user friendly the interface is (Qualtrics, 2019). The user testing method is considered to be the most significant and valuable approach. Since it affords undeviating information concerning how real users use and benefited from a framework; it illustrates exactly what problems users encountered in their interaction. (Nielsen and Mack 1994).

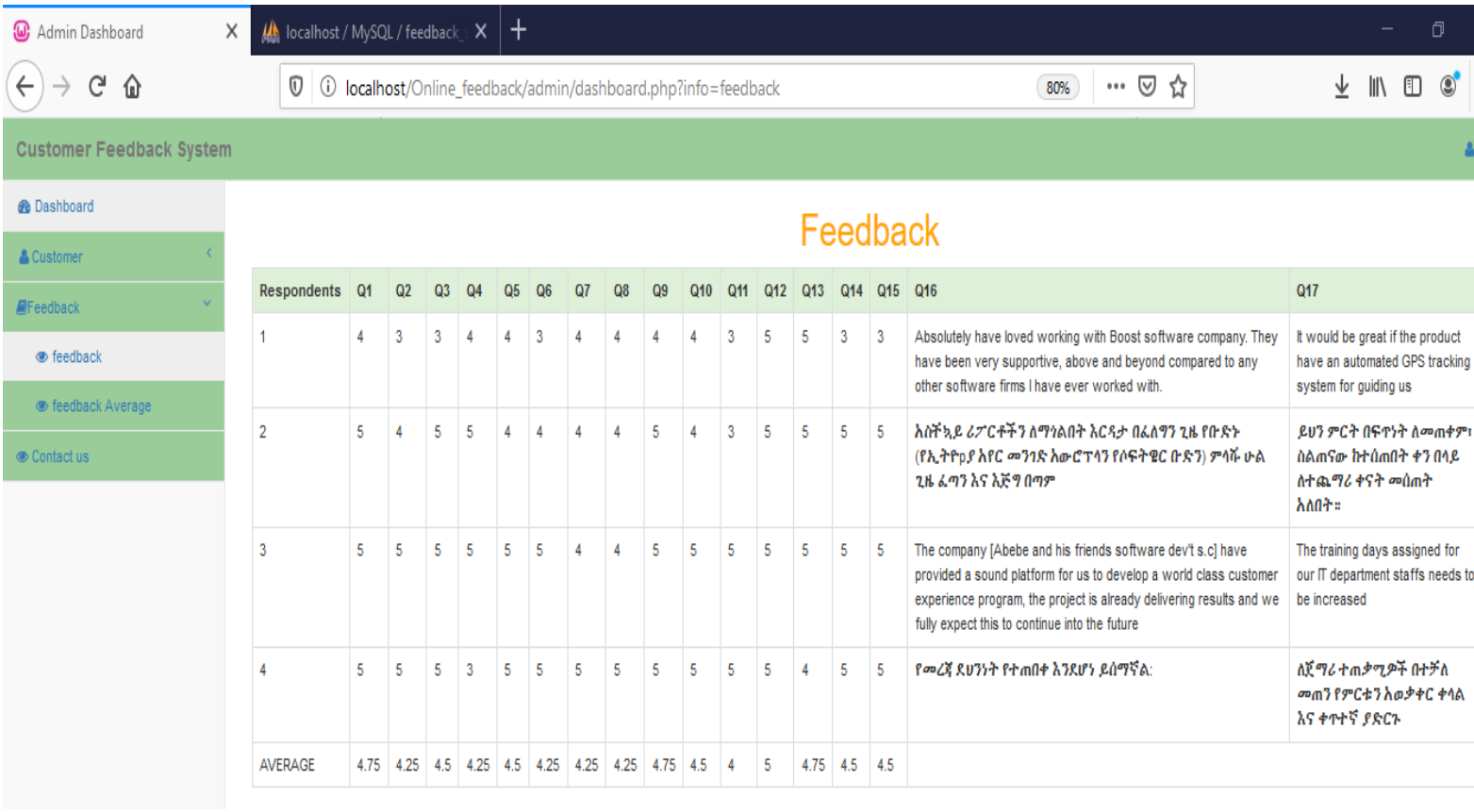
Dumas and Redish (1999) defined the user testing method as "a systematic way of observing actual users trying out a product and collecting information about the specific ways in which the product is easy or difficult for them". User testing materials were developed. These included: a testing script in order to welcome the users and to provide an introduction to the research; a consent form acknowledging the users'

agreement to participate in the test and to be observed through the testing session; a pre-test questionnaire to gather users' background information; a tasks scenario that included typical tasks for framework that was representative of actual use of the framework; and a post evaluation questionnaire to gather information about the preferences of users regarding the SPI framework after they had interacted with it.

Detail of customers participated in a feedback session

Four customers who are selected randomly from three firms have participated in the user acceptance testing with dual language English and Amharic (see Appendix 2). Among them one of them is customer of firm A, another is of firm C, and the rest two are from firm B. The evaluation takes place in the firms' office where access to their customers is easy and takes about 4 hours to perform. It consists of a set of typical tasks and involves completing questionnaires. A total of 4 end users were registered and give their feedback for the user acceptance testing. Data were gathered using a platform developed which is supported by a dual language features i.e. Amharic and English. A questionnaire which contain information developed based on usability factors organized into two categories as factors related with architecture and navigation of the product resulted from the framework and factors related with the overall customer experience including inconsistency, accessibility and customer service, architecture & design, navigation, missing functions, security and privacy, and content were used and observations of the users working through the products delivered and develop with the help of this SPI framework were made. Users provided a rank from 0–5 for each usability factor for the framework. The ranks were based on how easy or difficult it is to use the framework in specific scenarios. According to the users' response they classified themselves as very satisfied by rating the product on average 4.47 i.e. "extremely effective".

Analysis of customers' feedbacks



Respondents	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
1	4	3	3	4	4	3	4	4	4	4	3	5	5	3	3	Absolutely have loved working with Boost software company. They have been very supportive, above and beyond compared to any other software firms I have ever worked with.	It would be great if the product have an automated GPS tracking system for guiding us
2	5	4	5	5	4	4	4	4	5	4	3	5	5	5	5	አስቸኳይ ሪፖርቶችን ለማጎልበት እርዳታ በፈለግን ጊዜ የቡድኑ (የኢትዮpያ አየር መንገድ አውሮፕላን የሶፍትዌር ቡድን) ምላሹ ሁል ጊዜ ፈጣን እና እጅግ በጣም	ይህን ምርት በፍጥነት ለመጠቀም፣ ስልጠናው ከተሰጠበት ቀን በላይ ለተጨማሪ ቀናት መሰጠት አለበት።
3	5	5	5	5	5	5	4	4	5	5	5	5	5	5	5	The company [Abebe and his friends software dev't s.c] have provided a sound platform for us to develop a world class customer experience program, the project is already delivering results and we fully expect this to continue into the future	The training days assigned for our IT department staffs needs to be increased
4	5	5	5	3	5	5	5	5	5	5	5	5	4	5	5	የመረጃ ደህንነት የተጠበቀ እንደሆነ ይሰማኛል:	ለጀማሪ ተጠቃሚዎች በተቻለ መጠን የምርቱን አወቃቀር ቀላል እና ቀጥተኛ ያድርጉ
AVERAGE	4.75	4.25	4.5	4.25	4.5	4.25	4.25	4.25	4.75	4.5	4	5	4.75	4.5	4.5		

Figure 5.19 Analysis of received customer feedbacks

5.3.2 Result for the user acceptance testing

Table 5.1 Result for the user acceptance testing

Questions															Average	Percentage
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15		
4.8	4.3	5	4.3	5	4.3	4.3	4.3	4.8	4.5	4	5	4.8	4.5	4.5	4.4667	89.333333

Key:

Q1: I like the way this product looks

Q2: The convenient set-up of the product helps me find the information

Q3: I find the structure of the product clear

Q4: Icons/symbols used are appropriate

Q5: I find the terminologies/terms consistent

Q6: I find the structure of the product simple & straightforward

Q7: How would you rate your overall customer experience?

Q8: How satisfied were you with the product

Q9: How satisfied were you with customer Support

Q10: How satisfied were you with the timeliness of delivery

Q11: I am able to complete my work quickly using this product

Q12: The information provided with this product is clear

Q13: The product addressed my specific problems

Q14: Would you recommend our framework to be applied to others firms

Q15: I feel that the security of my information is protected

5.4 Discussion of the result

The framework results in a significant improvement of the firms' maturity of the software development process for firm A, 10.86% and 12%; firm B, 12.3% and 11%; and firm C, 7.11% and 9.5% improvements for CMMI V2.0 and DevOps maturity respectively. For the analysis of customers' feedback, 17 questions were prepared covering areas on architecture and navigation of the product developed with the help of the framework which mainly focuses on the look and feel aspects of the product, and the overall customer experience. Out of 17 questions, 15 of them were analyzed quantitatively in which the customer was marked its maturity on the rating scale 0 for not applicable and 1 to 5 accordingly. The rest two questions were answered qualitatively by specifying customers view on the product potency and flaws. According to the summary of customers' response gathered from users (see Table 5.1), the response was resulted on average to 4.47 points that the framework is very effective according to figure 5.18 and which indicate that customers are 89.3% satisfied by the product that they provided with the help of the framework. Even if the users found the structure of the product clear, the terminologies/terms to be consistent, and the information provided with this product is clear, they point out some major weaknesses to be well thought-out on the framework related with the considerations related with the set-up of the product towards helping find the information, structure, and icons/symbols used. To sum up, the framework designed is evaluated against the solution objectives and enables to cultivate a breakthrough improvement on the software development process of firms.

CHAPTER SIX

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This thesis research has tried to assess the extent of use (maturity) of the software processes and practices in the Ethiopian small software firms. Further, the research has provided bench mark data on the current status of the software practice in the firms for use in continuous assessment of future improvement efforts. The assessment of maturity level and improvement action plans for software process improvement programs often too awkward, time consuming, disruptive, and costly for small firms to implement since the cost of CMMI implementation was prohibitive for small firms. For small firms it is also hard to define, enact, formalize since it requires specializing skills about software processes, mastering special notations and tools. In order to support these firms CMMI V2.0 as a model for process improvement and DevOps as software development method were used. Firms which are initiating and implementing projects should look forward for better software process and practice maturity as empirical studies shows strong relation between process maturity level and meeting project goals. The second to third level of maturity is the suitable to describe this moment the maturity level in the organization. The study leads to the conclusion with the note that the current survey report supports the need for a framework that can bridge the existing gap in the maturity of the software process of the firms and highest matured firms of developed countries. Nevertheless, this thesis research is meant only a starting work only to assess the existing maturity situation of the Ethiopian small software firms and recommend some best software practices to be implemented in small firms' perception. To conclude, the framework results in a significant improvement of the firms' maturity of the software development process for firm A, 10.86% and 12%; firm B, 12.3% and 11%; and firm C, 7.11% and 9.5% improvements for CMMI V2.0 and DevOps maturity respectively. According to the research objective and the result of the research that has been conducted, it can be concluded that the potential users would prefer to use the SPI framework. The study doesn't apply the process metrics for the evaluation of the improvements measured at a project level like efficiency, productivity, cycle time, turnaround time, throughput, error rate, and cost effectiveness. The research presumed that future works will address these key process metrics for process evaluation.

6.2 Recommendations

The main goal is to do a starting work and open the door for further refinement and investigation and demonstrate the application of the concepts raised. This research work is a starting one and needs to be followed by a number of researches to investigate scopes which are not considered in this research and to refine the concepts raised and further enhance our understanding, and contribute to the software process improvement knowledge pool especially in small software firms. The research recommends that it is better to use as many firms as possible so as to cultivate deep information about the status of the firms with regards to process maturity by using process performance metrics, also known as KPIs (Key Performance Indicators) to measure how a process performs and reflect the accomplishment of individual goals in addition to maturity level assessment.

From the analysis of the current literature and also the assessment and final reports, the following recommendations are made to assist small firms undertaking SPI, and also assessors involved in such projects: before commencing SPI, ensure the firm is stable, firms should draw on expertise of external assessors/consultants as mentors, the SPI action plan should be realistically achievable within the evaluation time-scale, plan the evaluation from the start of the SPI program, this will be a source of motivation, and ensure that managers and development staff receive adequate training specific to the SPI model and areas of improvement.

It’s recommended that interested researchers can develop an enhanced SPI framework by combining machine learning techniques for predicting the software quality, parameters quantification that affects the quality, functionality and productivity of the software and process metrics with respect to product.

REFERENCE

ABS. (2002). AusStats: 8126.0 Information Technology, Australia. Retrieved from http://www.abs.gov.au/ausstats on April 2019.

Adnet. (2018). CMMI Overview. Retrieved from http://www.adnetcmm.com/cmmi-overview.html on March 2019.

Agrawal, M., & Chari, K. (2007). Software effort, quality, and cycle time: A study of CMM level 5 projects. IEEE Transactions on Software Engineering, 33(3), 145–156

Anon. (2018).DevOps: Principles, Practices, and DevOps engineer role. Retrieved from https://www.altexsoft.com/blog/engineering/devops-principles-practices-and-devops-engineer-role/. On June 12, 2019.

Arega Seyoum, Muhammed Aragie, Daniel Tadesse. (2016). Growth of Micro and Small Enterprises in Addis Ababa City Administration: A Study on Selected Micro and Small Enterprise in Bole Sub City. International Journal of Scientific and Research Publications (IJSRP), 6(1). 582-583.

A. P. Cater-Steel. (2002). Low-rigour, rapid software process assessments for small software development firms. Paper presented at the Australian Software Engineering Conference (ASWEC'04), Australia. 368-377.

B. Kitchenham, L. Pickard, and S. Pfleeger, (1995). Case studies for method and tool evaluation. IEEE Softw., 12 (4), 52–62.

Biro, M., and Messnarz, R. (2009). SPI experiences and innovation for global software development. Software Process Improvement and Practice, 14, 243–245.

Brodman, J. G., & Johnson, D. L. (1994). What small business and small organizations say about the CMM: Experience report? Proceedings of the 16th International Conference on Software Engineering (pp. 331–340). Los Alamitos: IEEE Computer Society Press.

Brodman, J. G., & Johnson, D. L. (1998). Applying the CMM for small organizations and small projects, proceedings of the software engineering process group conference. 8-9.

C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wessl ´ en. (2000). Experimentation in software engineering: an introduction. International Series in Software Engineering. Kluwer Academic Publishers, Norwell, USA,

Callanan, M. & Spillane, A. (2016). DevOps: Making It Easy to Do the Right Thing. IEEE Software 33(3): 53–59.

Carvalho, L., Scott, L., Jeffery, R. (2005). "An exploratory study into the use of qualitative research methods in descriptive process modelling", Information and Software Technology, pp. 113-127.

Casey, V., and Richardson, I. (2004). A Practical Application of the IDEAL Model. Software Process: Improvement and Practice 9(3), 123–132.

Cater-Steel, A., Toleman, M., and Rout, T. (2006). Process Improvement for Small Firms: An Evaluation of the RAPID Assessment-based Method. Information and Software Technology 48, 323–334.

Central Statistics Authority. (2003). Report on small scale manufacturing industries survey. Ethiopia, Addis Ababa: Central Statistics Authority.

Chevers, D. A., & Duggan, E. W. (2010).A preliminary study of the use of software process improvement initiatives in Jamaica. Proceedings of the 3rd International Conference on Information Resources Management, Montego Bay, Jamaica.

CMMI Institute Partner Workshop. (2019).CMMI V2.0. Available at https://www.slideshare.net/tranhuutan1/cmmi-v20retrieved on 20.06.2019.

CMMI Institute. (2019a). Overview of CMMI V2.0. available at https://www.umsec.umn.edu/sites/www.../Aug2018.CMMI%20V2.pdf retrieved on 20.06.2019.

CMMI Institute. (2019b).How Is CMMI V2.0 Different from V1.3?Available at https://cmmiinstitute.zendesk.com/hc/en-us/articles/360000175667-How-is-CMMI-V2-0-different-from-V1-3retrieved on 20.06.2019.

CMMI product team. (2010). CMMI for Development (CMMI-DEV), V1.3, CMU/SEI-2010th- TR-033 ed, Carnegie Mellon University. Pittsburgh, PA, USA.

CMMI® Product Team. (2002) “Capability Maturity Model Integration (CMMI®) for Software Engineering Version 1.1”, Software Engineering Institute, Carnegie Mellon University.

C. Hofer. (2002). Software development in Austria: results of an empirical study among small and very small enterprises. Paper presented at the 28th Euromicro Conference (EUROMICRO’02). Dortmund, Germany.

D. Azar, H. Harmanani, and R. Korkmaz. (2009). “A hybrid heuristic approach to optimize rule-based software quality estimation models,” Information and Software Technology, vol. 51, no. 9, pp. 1365–1376.

D. Jacobs. (2006). Accelerating Process Improvement Using Agile Techniques. New York: Auerbach.

David H. Nguyen. (2020). Three Ways for Scientists to Communicate Their Results of Scientific Research. Retrieved from https://sciencing.com/three-ways-for-scientists-to-communicate-their-results-of-scientific-research-12758603.html. on February 10,2020.

Dangle, K.C., P. Larsen, M. Shaw and M.V. Zelkowitz, (2005). Software process improvement in small organizations: A case study. IEEE Software, 22: 68-75.

Daskalantona, M. K. (1994). Achieving Higher SEI Levels. IEEE Software, Vol. 11, No. 4, pp. 17-24.

David Sarokin. (2019).Definition of small-scale enterprise retrieved from https://smallbusiness.chron.com/definition-smallscale-enterprise-17652.html. On 07/02/2019.

Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. (2016). DevOps. IEEE Software. 33, 94–100.

Eekels, J., and Roozenburg, N.F.M. (1991). A methodological comparison of the structures of scientific research and engineering design: their similarities and differences. Design Studies, 12(4), 197-203.

Eficode. (2016). DevOps Quick Guide. Online. Available at http://devops-guide.instapage.com/. Retrieved on April, 2019.

El Eman, K., Drouin, J.N., Melo, W. (1997) “SPICE the theory and practice of Software Process Improvement and Capability Determination”, IEEE Computer Society Press, Los Alamitos, California.

Elberzhager F, Arif T, Naab M, Süß I &Koban S (2017) From Agile Development to DevOps: Going Towards Faster Releases at High Quality – Experiences from an Industrial Context. Proc. 9th International Conference on Software Quality. Complexity and Challenges of Software Engineering in Emerging Technologies. Switzerland: Springer International Publishing: 33–44.

Erich, F. M. A., Amrit, C., and Daneva, M. (2017). A Qualitative Study of DevOps Usage in Practice. Journal of Software Evolution and Process 29(6), e1885. ISSN: 20477481.

Ethiopian IT professional’s association. (2020). INFORMATION TECHNOLOGY GROWTH IN ETHIOPIA. retrieved from http://www.eitpa.org/ on march 24, 2020.

Ferreira, A., Santos, G., Cerqueira, R., Montoni, M., Barreto, A., Rocha, A., Figueiredo, S., Barreto, A., Filho, R., Lupo, P., Cerdeiral, C. (2006). Taba Workstation: Supporting Software Process Improvement Initiatives Based on Software Standards and Maturity Models. In: Richardson, I., Runeson, P., Messnarz, R. (eds.) EuroSPI 2006. LNCS, 4257, 207–218. Springer, Heidelberg.

Galin, D. &Avrahami, M. (2006). Are CMM Program Investment Beneficial? Analysing Past Studies. IEEE Software, 23 (6). 81-87.
Git. (2019). A fast version control. Retrieved from https://git-scm.com/ on May 24, 2019.

Goldenson, D. R., &Herbsleb, J. D. (1995). After the appraisal: A systematic survey of process improvement, its benefits, and factors that influence success.SEI, 9,1–66.

Gorla, N., & Lin, S. (2010). Determinants of software quality: A survey of information systems project managers. Information and Software Technology, 52, 602–610

Habib, M., S. Ahmed, A. Rehmat, M. J. Khan and S. Shamail, (2008). Blending six sigma and CMMI-an approach to accelerate process improvement in SMEs. Proceedings of the 12th IEEE International Multitopic Conference, Karachi, Pakistan, 386-391.

Hendrik Ebbers. (2016). Maven vs. Gradle and the Best of Both Worlds. Retrieved from https://dzone.com/articles/maven-vs-gradle-and-the-best-of-both-worlds on March 24,2020.

Humble J and Farley D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston: Addison-Wesley Professional.

Humble, J., and Molesky, J. (2011). Why Enterprises Must Adopt Devops to Enable Continuous Delivery. Cutter IT Journal. 24, 6–12.
Humphrey, Watts. (1989). Managing the software process. Reading, MA.: Addison-Wesley.

Humphrey Watts. (1993). Introduction to Software Process Improvement. A technical report.

I. Richardson, (2002). SPI models: what characteristics are required for small software development companies? Software Quality, 10(2), 100-113.
I. Sommerville and J. Ransom, (2005). An empirical study of industrial requirements engineering process assessment and improvement, ACM Trans.Softw. Eng. Methodol., 14. 85-117

IEEE Computer Dictionary. (1991). Compilation of IEEE Standard Computer Glossaries, 610-1990, Institute of Electrical and Electronics Engineers Staff Institute of Electrical and Electronics Engineers, ISBN: 1559370793.

ISO, (2001). ISO Standard 9126: Software Engineering Product Quality, parts 1, 2 and 3, Std.

ISO, ISO 25000. (2005). Software Engineering Software Product Quality Requirements and Evaluation (SQuaRE), Std.

John A. Krosinick, Leandre R. Fabrigar. (1997). Designing rating scales for effective measurement in surveys. New York.

J. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, and M. Paulk, (1997). Software quality and the Capability Maturity Model. Commun. ACM, 40. 30-40.

J. P. Kuilboer and N. Ashrafi. (2000). Software Process and Product Improvement: An Empirical Assessment, Information and Software Technology, 42. 27-34.

J. Pearl. (1998). Why there is no statistical test for confounding, why many think there is, and why they are almost right, Department of Statistics, UCLA.

J. Roche. (2013). Adopting DevOps practices in quality assurance. Commun. ACM, 56(11):38-43.

Jabbari, R., Ali, N. bin, Petersen, K., and Tanveer, B. (2016). What is DevOps? A Systematic Mapping Study on Definitions and Practices. Proceedings of the Scientific Workshop Proceedings of XP 2016 on - XP '16 Workshops, 1–11. ISSN: 07421222.

Jiang, J. J., Klein, G., Hwang, H. G., Huang, J., & Hung, S. Y. (2004). An exploration of the relationship between software development process maturity and project performance.Information& Management, 41(3), 279–288.

Ken Peffers, TuureTuunanen, Marcus A. Rothenberger, Samir Chatterjee. (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information System, 24(3), 45-78.

Khan, M.I., M.A. Qureshi and Q. Abbas, (2010). Agile methodology in software development (SMEs) of Pakistan software industry for successful software projects (CMM framework). Proceedings of the International Conference on Educational and Network Technology, June 25-27, Qinhuangdao, China, pp: 576-580.

Komi-Sirviö, S. (2004). "Development and Evaluation of Software Process Improvement Methods", VTT Publications 535, Doctoral Thesis.

Krishnan, M. S., & Keller, M. I. (1999). Measuring process consistency: Implications for reducing software defects. IEEE Transactions on Software Engineering, 25(6), 769–781

Kromhout, B. (2017). Containers Will Not Fix Your Broken Culture (and Other Hard Truths). Queue. 15, 50:46–50:56.

Kunda, D., & Brooks, L. (2000). Assessing important factors that support component-based development in develop-ing countries. Information Technology for Development, 9, 123–139.

Laporte, C.Y., Alexandre, S., O'Connor, R.V. (2008a). A software engineering lifecycle standard for very small enterprises. In: R.V. O'Connor, N. Baddoo, K. Smolander, R. Messnarz (eds.) Software Process Improvement, no. 16 in Communications in Computer and Information Science, pp. 129–141. Springer, Berlin Heidelberg.

Laporte, C.Y., Renault, A., Alexandre, S. (2008b). The application of international software engineering standards in very small enterprises. In: H. Oktaba, M. Piattini (eds.) Software process improvement for small and medium enterprises techniques and case studies Information Science Reference, Hershey, New York. 42–70.

Lucy Ellen Lwakatare, (2017). DevOps adoption and implementation in software development practice concept, practices, benefits and challenges. (PhD dissertation).

Lukasiewicz, K. and J.Miler, (2012). Improving agility and discipline of software development with the Scrum and CMMI. IET Software, 416-422.

Manoj Kumar, and SohanLal. (2012). A Structure to Improve Software Process Improvement Model for Small Industry. Pacific University, Udaipur, Rajasthan, India. International Journal of Computer Science and Technology. 462-463.

Mark C. Paulk (1998). Using the software CMM in small organizations. Institute of Software Research, Paper 5 (1–13). Pittsburgh, PA: Software Engineering Institute.

Masters, S., and C. Bothwell. (1995). A CMM appraisal framework, version 1.0 (CMU/SEI-95-TR-001). Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

McCaffery, F., Taylor, P.S., Coleman, G. (2007). Adept: A unified assessment method for small software companies. IEEE Software 24(1), 24–31.

Miles, M. &Huberman, A. (1999). Qualitative Data Analysis. London: Sage.

Mohamed Sami, (2018). "The Software Process Improvement (SPI) – Reward or Risk," in Mohamed Sami - Personal blog. Retrieved May 10, 2019, from https://melsatar.blog/2018/06/26/the-software-process-improvement-spi-reward-or-risk/

Montoni, M., Santos, G., Rocha, A., Weber, K., de Araujo, E. (2007). MPS Model and TABA

M. Rossi. and M. K. Sein. (2001). Design Research workshop: A Proactive Research Approach." 26th Information Systems Research Seminar in Scandinavia, Haikko, Finland.

M. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren. (2011). Action Design Research. MIS Quarterly 35(1): 35-56.

Niazi, M., Wilson, D. &Zowghi, D. (2003). A maturity model for the implementation of software process improvement: an empirical study. The Journal of Systems and Software, 74(2). 155-172.

Niazi, M., & Babar, M. A. (2009). Identifying high perceived value practices of CMMI level 2: An empirical study. Information and Software Technology, 51, 1231–1243.

Niazi, M., Babar, M. A., &Verner, J. M. (2010). Software process improvement barriers: A cross-cultural comparison. Information and Software Technology, 52, 1204–1216

Niazi, M. (2012). An exploratory study of software process improvement implementation risks.Journal of Software: Evolution and Process, 24, 877–894.

Oktaba, H. (2006). MoProSoft: A Software Process Model for Small Enterprises. In: Proceedings of the First International Research Workshop for Process Improvement in Small Settings, pp. 93–101. Carnegie Mellon University, Pittsburgh.

Oktaba, H., Garcia, F., Ruiz, F., Pino, F. J., &Alquicira, C. (2007). Software process improvement: The competisoft project. IEEE Computer Society, 40,21–28.

Olsson H, Alahyari H & Bosch J (2012) Climbing the" Stairway to Heaven"–A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. Proc. 38th Euromicro Conference on Software Engineering and Advanced Applications. IEEE: 392–399.

P. Savolainen, H. M. Sihvonen, and J. Ahonen. (2007). "SPI with lightweight software process modeling in a small software company", Lecture Notes in Computer Science, 4764. 71-81.

Paulk, M. C., Weber, C. V., Curtis, B., &Chrissis, M. B. (1995). The capability maturity model: Guidelines for improving the software process. Reading, MA: Addison Wesley Longman, Inc.

Pfleeger, S. L. & Atlee, J. M., (2006). Software Engineering: Theory and Practice, 3rd. ed. New Jersey, USA: Pearson Prentice Hall.

Pino, F. J., Pardo, C., Garcia, F., &Piattini, M. (2010). Assessment methodology for software process improvement in small organizations. Information and Software Technology, 52, 1044–1061.

Pino, F.J., García, F. and Piattini, M. (2009). An Integrated Framework to Guide Software Process Improvement in Small Organizations. In: O'Connor, R.V., Baddoo, N., Gallego, J.C., Muslera, R.R., Smolander, K., Messnarz, R. (eds.) EuroSPI 2009. CCIS,42. 219–224. Springer, Berlin.

Puppet Labs (2015). State of Devops 2015 Report. IT Revolution Press. Online. Available at: https://puppet.com/resources/white-paper/2015-state-of-devops-report. Retrieved on 11.04.201.

Qualtrics. (2019). Customer Satisfaction (CSAT) Surveys: Examples, Definition & Template. https://www.qualtrics.com/experience-management/customer/satisfaction-surveys/

Rimawi, Y., Amescua, A., Cuevas, G., San Feliu, T., Garcia, J.: Ramala. (2005). A Knowledge Base for Software Process Improvement. In: Proceedings of the Second International Conference on Innovations in Information Technology, Dubai, UAE.

Robillard, P. N., Kruchten, P. &d'Astous, P. (2003). Software Engineering Process. s.l.:Person Education.

R. Pressman. (2009) Software Engineering: A Practitioner's Approach. 7th international edn, McGraw-Hill Education, Newyork, USA.

Sadhana Deshpande, Valentine Casey, Ita Richardson, and Sarah Beecham. (2010). Culture in Global Software development - a Weakness or Strength? International Conference on Global Software Engineering. 67-68.

S. Birisci, M. Mentin, and M. "Karakas. (2009). Prospective Elementary Teacher's Attitudes Toward computer and internet use: A Sample from Turkey", World Applied Sciences Journal, .6(10), pp. 1433-1440.

Scott, L., Jeffery, R., Carvalho, L., D'Ambra, J., & Rutherford, P. (2001). Practical software process improvement—The IMPACT project. Proceedings of the Australian Software Engineering Conference, IEEE Computer Society. Los Alamitos, CA: IEEE Computer Society Press. 182–189

Seleniumhq. (2019). What is selenium. Retrieved from https://www.seleniumhq.org/ on may 28, 2019.

Sharmistha Kar,Satyabrata Das , Amiya Kumar Rath , and Subrata Kumar Kar. (2012). Self-assessment Model and Review Technique for SPICE: SMART SPICE. Conference in Communications in Computer and Information Science.

Sigurd Thunem. (1997). Process Modeling for Process Improvement - A Process Conformance Approach. MSc Thesis.

Software Engineering Institute (SEI). (2010). CMMI for development, version 1.3. Carnegie Mellon University.

Solomon Assefa. (2017). Implementation of Project Management Principles, Tools and Techniques at Ministry of Information Communication Technology. A Master's Thesis, School of Graduate Studies, St. Mary's University. 17-18.

Sommerville, I. (2011). Software Engineering, 9th ed. Harlow, England: Person Education Limited.

Staples, M., &Niazi, M. (2008). Systematic review of organizational motivations for adopting CMM-based SPI. Information and Software Technology, 50, 605–620.

Statistics Solutions. (2019). Paired Sample T-Test. Retrived from https://www.statisticssolutions.com/manova-analysis-paired-sample-t-test/ on 02/09/2019.

Stelzer, D. &Mellis, W. (1998). Success Factors of Organizational Change in Software Process Improvement. Software Process Improvement and Practice, 4(4). 227-250

Swartout, P. (2014). Continuous Delivery and DevOps: A Quickstart guide, Second Edition. Second Edi. Birmingham, UK: Packt Publishing, 1–196. ISBN: 9781849693684.

S. B. Basri, and R. V. O'Connor. (2011).Knowledge Management in Software Process Improvement: A Case Study of Very Small Entities. Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications. 273.

S. March and G. Smith. (1995). Design and Natural Science Research on Information Technology. Decision Support Systems 15: 251266.

Teka Degif, Dittrich Yvonne, Kifle, Mesfin. (2016). Usability challenges in an Ethiopian software development organization. 114-120.

Trudel, S., Lavoie, J. M, ParÈ, M. C. &Suryn, W. (2006). The small company dedicated software process quality evaluation method combining CMMI and ISO/IEC 14598. Software Quality Journal, 4(1). 7-23.

Turner Consulting Group, (2007). CMMI mentoring service. Retrieved from http://www.tcg.com/services-CMMI-mentoring-overture.html?OVRAW=elements%20

Udacity. (2016a). Definition of DevOps [Video File]. Retrieved from https://youtu.be/p41OYYbatgQ.

Udacity. (2016b). A is for Automation [Video File]. Retrieved from https://youtu.be/n7K5SpMrQt0.

Udacity. (2016c). M is for Measurement [Video File]. Retrieved from https://youtu.be/4IXgVwxhnvU.

Udacity. (2016d). S Stands for Sharing [Video File]. Retrieved from https://youtu.be/bC4Gyyzu7K8.

VepambattuChandu. (2019). Top five reasons why DevOps is important. Retrieved from https://dzone.com/articles/top-5-reasons-why-devops-is-important on May 14, 2019.

Wangenheim, C.G., Anacleto, A., and Salviano, C.F. (2006). Helping Small Companies Assess Software Processes. IEEE Software 23(1), 91–98.

Weber, K., Araújo, E., Rocha, A., Machado, C., Scalet, D., Salviano, C. (2005). Brazilian Software Process Reference Model and Assessment Method. In: Yolum, p., Güngör, T., Gürgen, F., Özturan, C. (eds.) ISCIS 2005. LNCS, 3733. 402–411. Springer, Heidelberg.

Weick, K. (1995). What Theory Is Not, Theorizingis Quarterly,40(1).385-390.

Wieringa, R. (2014). Design science methodology for information systems and software engineering. Berlin: Springer.Workstation: Implementing Software Process Improvement Initiatives in Small Settings. In: Proceedings of the Fifth International Workshop on Software Quality (WoSQ'07), p. 4. IEEE Computer Society, Washington.

Xu Lai &SjaakBrinkkemper. (2007). "Concepts of Product Software: Paving the Road for Urgently Needed Research," Technical report, Institute of Information and Computing Sciences, Utrecht University, The Netherlands. European Journal of Information Systems 16, 531–541.

Zhang, L. and D Shao, (2011). Software process improvement for small and medium organizations based on CMMI. pp:22-25.

Zubrow, D., W. Hayes, J. Siegel, and D. Goldenson. (1994). Maturity questionnaire (CMU/SEI-94-SR-7). Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

APPENDICES

Appendix 1 Questionnaire

Dear Respondent:

The student at Bahir Dar University Software Engineering department is conducting a research on software process improvement framework development for small scale software developing firmsfor the fulfillment of his MSc Degree. The research will investigate the following issues:

1. Whether and to what extent each of the practices of the software process improvement are being applied in performing software projects.
2. The areas that need focus for development and improvement of the firms' software processes.

The research output will provide a framework that will support for software process improvement by considering practice areas categories: doing, managing, enabling, and improving. In addition, it also provides information about process maturity considering all DevOps focus areas: culture, automation, measurement, and sharing.

Your precious time and effort in participating in this research will also contribute to the development and improvement of the process in your company. Thus, you are kindly invited to fill out the questionnaire and return to the researcher.

Thank you for your interest in participating in the research.

Sincerely yours

Tadele Mengiste

Mobile … +251918651955

Questionnaire Survey for MSc Thesis on the Software Process Improvement framework development for small scale software developing firms.

Part one- General Information

Direction: Please provide the required information on the space provided

1. Name of the person filling the questionnaire (*Optional*)----------
2. Position/role in the company----------
3. How many employees are there in your company currently?
4. How long have you been with the company?

Part two: Development process questions

1. What development methodology/process does your present office use?

2. What development tools and programming language does your office use?

3. Are you satisfied with this development methodology/process? Yes ☐ No ☐
4. Could you please briefly highlight on the strengths and weaknesses of your present methodology/process.

5. Do you have any experience on CMMI frameworks? Yes ☐ No ☐
6. If yes what will you consider as CMMI frameworks strength and weakness?

7. What do you think about CMMI frameworks in Ethiopian software industry?
    a. Does it improve software quality? Yes ☐ No ☐
    b. Would you recommend it for your company? Yes ☐ No ☐
8. Do you have any experience in DevOps or other agile family development methodology? Yes ☐ No ☐
9. If yes what will you consider as DevOps strengths and weakness?

10. Do you think there is any relationship between CMMI frameworks and DevOps practices? Yes ☐ No ☐
    a. If yes, what are the relationships?

    b. If no, please explain briefly.

11. Will you like try out DevOps if given the opportunity? Yes ☐ No ☐
12. Are you satisfied with software development process in Ethiopia? Yes ☐ No ☐
13. What other issues would you want to be addressed regarding development methodologies in Ethiopian market?

14. Have you received any process improvement related training? yes ☐ No ☐

    If yes, what was the highest level of training you have received?

    A. Masters level B. Bachelors C. Certificate

    D. Short - term training E. As a course in related program of study

    F. Others (*Specify*)_____

15. Which of the following best describe your current position?

    Project Manager ☐ Software Tester ☐ Web Designer ☐

    Programmer ☐ Requirement Engineer ☐ System Analyst ☐

    Others (*Specify*)_____

16. Have you applied any framework for an improvement of the software process in your firm? Yes ☐

    No ☐

    If yes, which software Process Improvement Model/ Framework is being applied in your firm?

    CMMI ☐ SPICE ☐ TickIT ☐

    Bootstrap ☐ TSP and PSP ☐ Others (*Specify*)_____

Part Three –Software Process Measurement Questions

Instruction: Select one project that you have performed before and try to react the following questions accordingly.

1. Write the name of the project that you have developed earlier if possible (*Optional*).
2. What was the nature/ application domain of the project that you have developed? E.g. embedded system, information system ...etc.
3. What tools and languages were you used?
4. How long does it take to finish the project?
5. How many individuals participated during the project?
6. How much money do you invest till the project finished?

Part Four –Software Process Maturity Questions

General direction

Answer the following questions based on your knowledge of software process improvement in the project that you are participating or in the firm you are working. Please choose the ascending maturity level one up to five based on the key characteristics which were taken from capacity maturity model integration version 2.0 (CMMI V2.0) of Software Engineering Institute.

- ✓ Maturity Level 1: **Initial**
  Unpredictable and reactive. Work gets completed but is often delayed and over budget.
- ✓ Maturity Level 2: **Managed**
  Managed on the project level. Projects are planned, performed, measured, and controlled.
- ✓ Maturity Level 3: **Defined**
  Proactive, rather than reactive. Organization-wide standards provide guidance across projects, programs, and portfolios.
- ✓ Maturity Level 4: **Quantitatively Managed**
  Measured and controlled. Organization is data-driven with quantitative performance improvement objectives that are predictable and align to meet the needs of internal and external stakeholders.
- ✓ Maturity Level 5: **Optimized**
  Stable and flexible. Organization is focused on continuous improvement and is built to pivot and respond to opportunity and change. The organization's stability provides a platform for agility and innovation.

Maturity Level in 4 Categories, 12 Capability Areas, and 39 practice areas of CMMI Dev 2.0

N.A= Not applicable

No.	Software Process areas Key Practice Characteristics	Maturity Levels						Remarks
		1	2	3	4	5	0	
		Initial	Managed	Defined	Quantitatively Managed	Optimized	N.A	
1	**Doing**							
	1.1 Does the project follow a written organizational policy for managing the system requirements allocated to software?							
	1.2 Do you have any quality assurance activities for software projects?							
	1.3 Do the software evaluated during or at the end of the development process to determine whether it satisfies specified business requirements?							
	1.4 Does the firm prepare and perform peer reviews on selected work products using established procedures.							
	1.5 Does the firm follow a formal procedure to help in the selection of the design and implementing solution to requirements?							
	1.6 Does the firm follow a defined integration strategy and procedures to achieve complete product integration through progressive assembly of product components, in one stage or in incremental stages?							

	1.7 Does the firm follow a procedure for monitoring all aspects of planning, communicating, managing, deploying, and confirming that service system components effectively make the transition to the delivery environment?							
	1.8 Does the firm follow a formal procedure to analyze existing service agreements and service data to prepare for updated or new agreements?							
	1.9 Do the firm follows a strategic service management processes to improve alignment between the set of services offered by a service provider organization and its strategic business objectives?							
	1.10 Does the firm identify qualified potential suppliers and distribute the solicitation package for their response?							
	1.11 Do you determine the methods to be used in the purchasing of a product and product components?							
2	**Managing**							
	2.1 Do you estimate size, effort, and cost for software projects?							
	2.2 Does the firm follow a defined procedure to develop a list of tasks and assign people to tasks?							
	2.3 Do you determine the methods to be used in the tracking of actual results against estimates for size, effort, schedule, resources, knowledge and skills, and budge?							

	2.4 Does the firm monitor, analyze, understand, and report on current and future demand for services, use of resources, capacity, service system performance, and service availability?							
	2.5 Does the firm follow a risk management plan for monitoring identified risks or opportunities and communicate status to affected stakeholders?							
	2.6 Is an approach for incident resolution and prevention developed, keep updated, and followed?							
	2.7 Does the firm develop contingency approaches for managing significant disruptions to operations?							
	2.8 Does the firm consider employee compensation and reward mechanism *to* provide all individuals with remuneration and benefits based on their contribution and value to the organization?							
	2.9 Does the firm use a formal procedure for staffing and workforce management?							
	2.10 Does the firm use a career development strategy to ensure that individuals are provided opportunities to develop workforce competencies that enable them to achieve career objectives?							
	2.11 Do members of the software engineering group and other software-related groups receive the training necessary to perform their roles?							
	2.12 Does the firm invest workgroups with the responsibility and authority for determining how to conduct their business activities most effectively?							

3	**Enabling**							
	3.1 Does the firm use a formal procedure for managing and planning security?							
	3.2 Does the firm develop secure solutions/products?							
	3.3 Does the firm follow a formal procedure to manage security threats and vulnerabilities?							
	3.4 Does the firm follow a defined strategy and procedures to select and manage secure suppliers?							
	3.5 Does the firm plan and support security in work?							
	3.6 Does the firm determine root causes of selected outcomes by following an organizational process?							
	3.7 Does the firm analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria?							
	3.8 Do you have any general configuration management policy of the institution to keep updated, and use records describing and managing changes in the items?							
	3.9 Does the firm establish timely communication across the organization and ensure that the workforce has the skills to share information and coordinate their activities efficiently?							
	3.10 Does the firm follow a formal procedure for managing and planning safety?							

	3.11 Does the firm follow a formal procedure for ensuring safety?							
4	**Improving**							
	4.1 Does the senior management identifies what is important for doing the work and defines the approach needed to accomplish the objectives of the organization?							
	4.2 Do you have any activities for providing sufficient resources, funding, and training for developing and performing processes?							
	4.3 Does the firm provide a standard procedure for exploring and evaluating potential new processes, techniques, methods, and tools to identify improvement opportunities?							
	4.4. Does the firm determine what process assets will be needed to perform the work and make these assets available for use?							
	4.5 Are you giving a course to the people regarding statistical methods, data collection, analysis and reporting processes?							

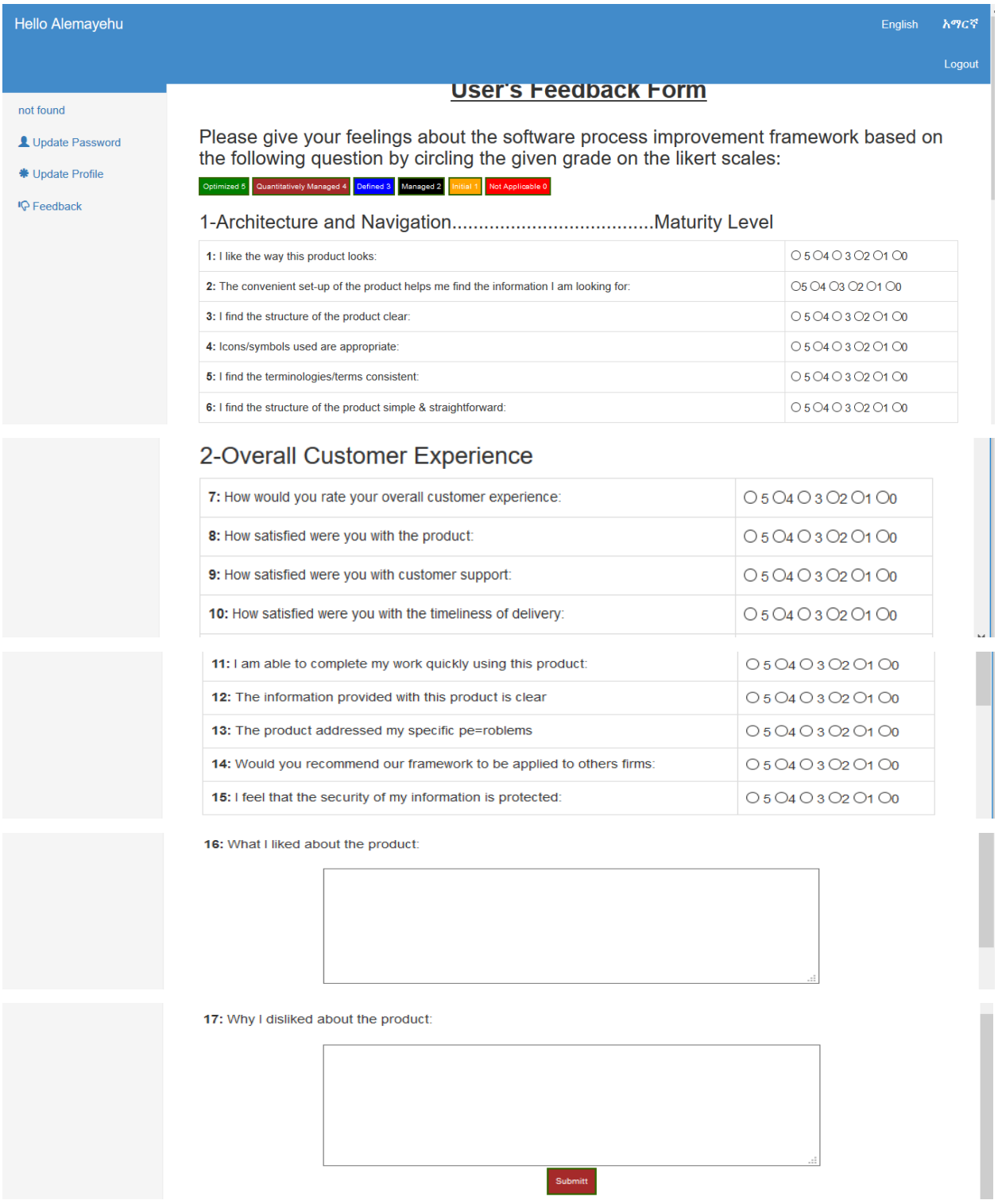
Maturity Level in 4 DevOps Dimensions

No.	DevOps Focus Areas	Maturity Levels						Remarks
		1	2	3	4	5	0	
		Initial	Managed	Defined	Quantitatively Managed	Optimized	N.A	
1	**Culture**							
	1.1.How aware are your employees of core DevOps principles and their benefit for your organization?							
	1.2. How skilled is your organization in DevOps methods and practices?							
	1.3.How incentivized and motivated is your workforce to apply DevOps practices?							
	1.4.How much the teams in the firm are customer and product oriented?							
	1.5.How much do you cultivate an environment of continuous learning?							
2	**Automation**							
	1.1.How mature is your release and deployment automation?							
	1.2.How mature is your test automation?							
	1.3.How mature are your build and integration processes?							
	1.4.How automated is your environment provisioning process?							
	2.5 How automated is configuration of your system?							
3	**Measurement**							

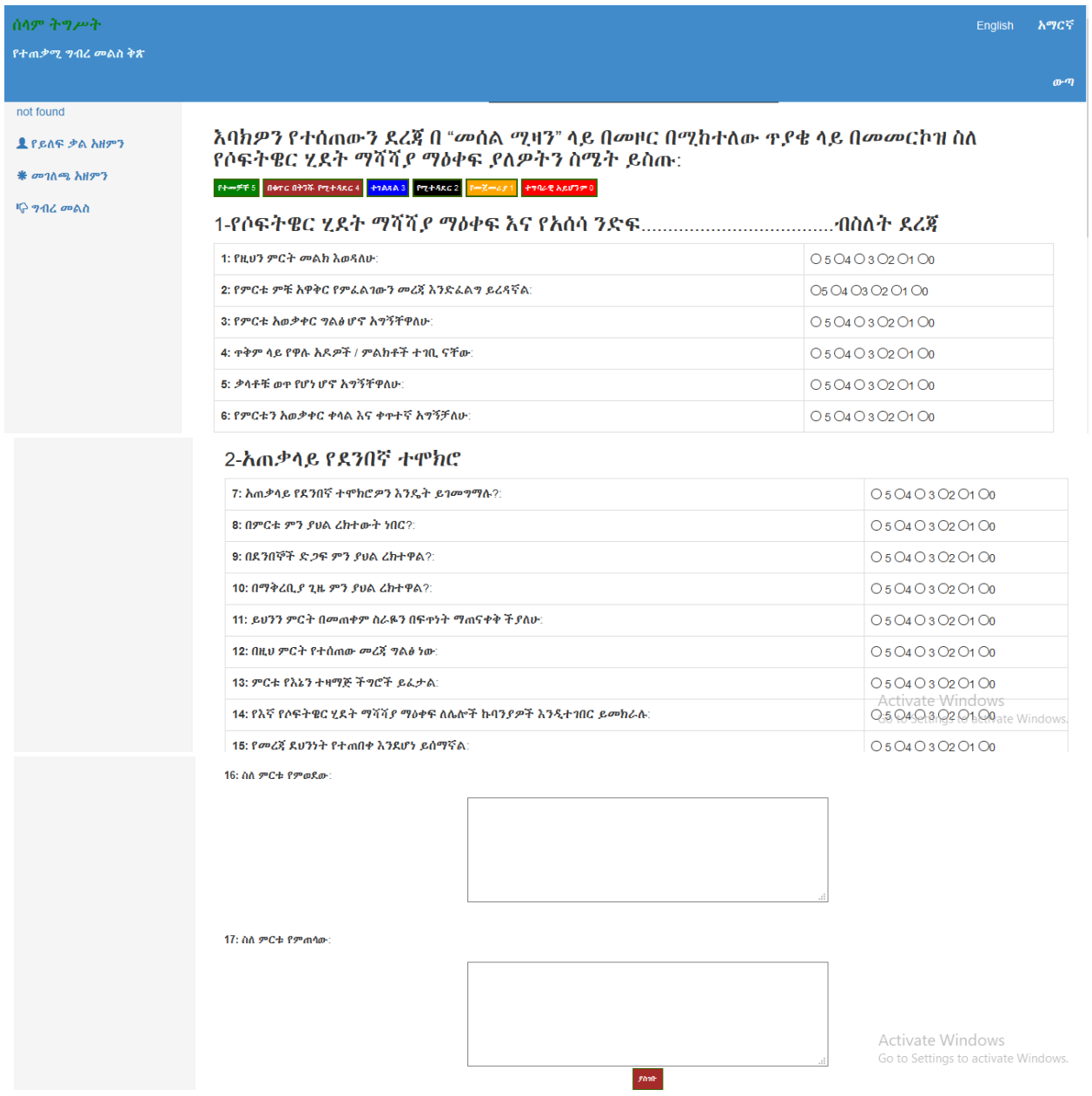
	3.1. How plans are created with features that are tied directly to customer feedback with metrics defined to ensure customer expectations are met?							
	3.2. Does a continuously updated dashboard is in place to show progress of features?							
	3.3. To what extent does the firm monitor high-level business metrics such as revenue or end-to-end transactions per unit time?							
	3.4 Does the firm use a procedure on how many people are using your product right now?							
	3.5 To what extent does the firm measures how long did it take to go from development to deployment?							
4	**Sharing**							
	4.1. To which extent is you align and share your processes between Dev & Ops?							
	4.2. How do development and operations teams collaborate during a production issue?							
	4.3. How strongly do you collaborate to building shared knowledge?							
	4.4 Do all the members of your product team have access to code, status, metrics and history?							
	4.5 To what extent do the potential changes visible to all members of your product team?							

Appendix 2 Customer Satisfaction Assessment Forms

English Version of the customer satisfaction assessment



Amharic Version of the customer satisfaction assessment



Appendix 3 Assessment Result

Firm A								
CMMI V2.0 Maturity Level								
	Doing		Managing		Enabling		Improving	
Respondents	Before	After	Before	After	Before	After	Before	After
1	35	39	40	39	38	40	13	21
2	33	44	37	43	30	43	20	18
Average	34	41.5	38.5	41	34	41.5	16.5	19.5
Percentage(%)	61.82	75.45	64.17	68.33	61.82	75.45	66	78
Total Mean(%)	63.45075758				74.31060606			

Firm B								
CMMI V2.0 Maturity Level								
	Doing		Managing		Enabling		Improving	
Respondents	Before	After	Before	After	Before	After	Before	After
1	30	44	36	46	37	31	16	22
2	32	46	41	48	32	42	15	17
3	33	41	38	46	34	39	15	13
Average	31.6667	43.6667	38.3333	46.6667	34.3333	37.3333	15.3333	17.3333
Percentage(%)	57.58	79.39	63.89	77.78	62.42	67.88	61.3333	69.3333
Total Mean(%)	61.30555556				73.5959596			

Firm C								
CMMI V2.0 Maturity Level								
	Doing		Managing		Enabling		Improving	
Respondents	Before	After	Before	After	Before	After	Before	After
1	35	42	40	37	38	38	13	21
2	42	46	42	39	37	43	16	17
Average	38.5	44	41	38	37.5	40.5	14.5	19
Percentage(%)	70.00	80.00	68.33	63.33	68.18	73.64	58	76
Total Mean(%)	66.12878788				73.24242424			

==========

Firm A								
DevOps Maturity Level								
	Culture		Automation		Measurement		Sharing	
Respondents	Before	After	Before	After	Before	After	Before	After
1	17	16	21	24	14	18	16	18
2	15	21	18	21	14	14	15	22
Average	16	18.5	19.5	22.5	14	16	15.5	20
Percentage(%)	64.00	74.00	78.00	90.00	56.00	64.00	62.00	80.00
Total Mean(%)	65				77			

Firm B								
DevOps Maturity Level								
	Culture		Automation		Measurement		Sharing	
Respondents	Before	After	Before	After	Before	After	Before	After
1	10	18	14	16	22	23	19	23
2	22	22	17	19	21	23	16	20
3	19	21	20	21	20	22	17	22
Average	17	20.3333	17	18.6667	21	22.6667	17.3333	21.6667
Percentage(%)	68.00	81.33	68.00	74.67	84.00	90.67	69.33	86.67
Total Mean(%)	72.33333333				83.33333333			

Firm C								
DevOps Maturity Level								
	Culture		Automation		Measurement		Sharing	
Respondents	Before	After	Before	After	Before	After	Before	After
1	18	18	20	23	14	18	16	18
2	17	21	21	20	19	23	18	21
Average	17.5	19.5	20.5	21.5	16.5	20.5	17	19.5
Percentage(%)	70.00	78.00	82.00	86.00	66.00	82.00	68.00	78.00
Total Mean(%)	71.5				81			

==========

CMMI V2.0 Process Improvement			
	Firm A	Firm B	Firm C
Before	63.45	61.31	66.13
After	74.31	73.5959596	73.24

DevOps Process Improvement			
	Firm A	Firm B	Firm C
Before	65.00	72.33	71.50
After	77.00	83.33333333	81.00