Software Quality Assurance in Agile and Waterfall Software Development Methodologies: A Gap Analysis.

BY

Lakmali De Zoysa

(2009/MISM/05)

Submitted in accordance with the requirements for the degree of

MASTERS IN INFORMATION SYSTEMS MANAGEMENT

at the

UNIVERSITY OF COLOMBO

SUPERVISOR: DR. KAPILA PONNAMPERUMA

CO-SUPERVISOR: DR GAMINI WIJERATNE

FEBRUARY 2011

DECLARATION

I certify that this Dissertation does not incorporate without acknowledgement any material previously submitted for the Degree or Diploma in any University, and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text

Date: ....

Lakmali De Zoysa

The undersigned, have supervised the dissertation entitled **UALITY ASURANCE IN AGILE AND WATERFALL SOFTWARE DEVELOPMET METHODOLOGIES: A GAP ANALYSIS** presented by Lakmali De Zoysa, a candidate for the degree of Masters in Information Systems Management, and hereby certify that, in my opinion, it is worthy of submission for examination.

Date: ....

Dr. Kapila Ponnamperuma

Supervisor

ACKNOWLEDGMENT

It is a pleasure to thank those who made this thesis possible. I am heartily thankful to my supervisors, Dr Kapila Ponnamperuma, Dr. Gamini Wijayarathne and our coordinator Dr. Chaminda Jayasundara, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject. Also I owe my deepest gratitude to Dr. Mrs. Kodikara Librarian and Dr. Ruwan Gamage Assistant Librarian of the University Moratuwa for kindly granting me permission to access the thesis section of the library. I'm very much grateful to Mrs. Rukie Salgado and my uncle Mr. Arunashantha De Silva Deputy Legal Draftsmen for the government of Sri Lanka, for their eagerness to correct the language of the document. This thesis would not have been possible if not for the enormous support given in data collection by Mr. Wasantha Kumara, Mr. Kanishka Karunarathne, Mr. Tharaka Kulasinghe, Mrs. Nimeka Abedeera, Mr. Gayath Chaminda, Mr. Amila Dharmaratne, Mr. Ruvinda Rathnegoda and Mr. Chinthaka Indikadahena. I'm grateful to Mr. Rukshan Abeygunawardene lecturer Statistics Department of the University of Colombo and Mrs Nadeeja Dodangoda, lecturer American College of Higher Education, without whose guidance, support and encouragement the analysis of the research would have been such a difficult task. I would like to thank my batch mate Mr. Naleen Perera, for sharing his knowledge throughout the process and the guidance given on handling SPSS software. Words alone cannot express the thanks I owe to my parents, my grand aunt and my husband, for their constant encouragement and assistance so readily given. Lastly, I offer my regards and blessings to all those who rendered their support in any way during the completion of this project.

ABSTRACT

The purpose of this study was to investigate whether the software development companies can achieve expected software quality through Agile development. In order to reach this goal, the first objective of the research was to identify the software quality factors through various quality models and quality management philosophies. Secondly, to identify the software development process models. Thirdly, to analyse the software quality difference between development methodologies in terms of selected quality factors. And finally to identify the development technique by which high quality software products could develop. The research was conducted in the Sri Lankan context focusing on software development companies registered with the Sri Lanka Exports Association. After the preliminary investigation on obtaining relevant information, four companies namely; Virtusa, Team Work, DMS and E- College were selected for the research. The second pilot survey reflected that it was impossible to collect data from clients. Thus, the research was aimed only on developer oriented quality factors. These selected factors include correctness, testability, changeability, install ability, time and budget. The research was confined to analyse the quality difference only between Waterfall and Agile since the second pilot survey revealed that the Waterfall is the most widely used in Sri Lanka. Both qualitative and quantitative research methodology was utilized in this study. The researcher tendered 190 questionnaires among testers, developers and QA leads via e-mails. The response rate for questionnaires was eighty one percent and the accepted rate was seventy two percent. Twelve interviews were carried out with the Project Managers to capture project related information. The result of the questionnaire revealed no significant difference between the two development methods in achieving correctness and install ability. Whereas, the difference in testability and changeability was significant and reflected that Agile is better than Waterfall. The cumulative analysis of the product quality factors showed that a high level of software quality can be achieved through Agile development. Analysis of the interviews reflected that there is no significant difference in the software project quality between the two development methods. The author recommends applying Agile techniques for software development projects where the requirements are complex, difficult to capture and frequently fluctuating (At situations where high degree of testing and changeability is required). Any method can be used if the main focus is to achieve only project quality.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AM	Agile Modeling
ANSI	American National Standard Institute
ASD	Adaptive Software Development
CMMI	Capability Maturity Model Integration
FRUPS	Functionality, Reliability, Usability, Performance, Supportability
ICTA	Information and Communication Technology Agency
IID	Iterative Incremental Development
IMF	International Monitory Found
IT	Information Technology
PM	Project Manager
QA	Quality Assurance
RUP	Rational Unified Process
SAD	Systems Analysis and Design
SDLC	Software Development Life Cycle
SEA	Sri Lanka Exports Association
SLASI	Sri Lanka Association for the Software Industry
SLASSCOM	Sri Lanka Association of Software & Service Companies
SQA	Software Quality Assurances
SRS	Software Requirement Specification
SW	Software
SWD	Software Development
UP	Unified Process
XP	Extreme Programming

CHAPTER ONE

1. INTRODUCTION

1.1 OVERVIEW

One of the pioneers in software development arena Barry Boehm stated that "The last decade of the 20th century has witnessed a growing use of software products in a variety of application areas and their correct operation is significant for business success" (Boehm, 2006, pp. 20-21). Hence, Software development companies have to deliver quality software products within a shorter period. This situation has lead computer specialists, analysts and developers to introduce effective and efficient development models, as well as proven software project, process and product quality techniques (Boehm, 2006, pp. 24-25).

As per the Juran and Frank's definition, it is not only the customer who benefits from a focus on high quality but businesses that value quality become more responsive and innovative, increase their competitive differentiation, and greatly reduce their total cost of development and time to market (Juran & Frank, 1988). If a company unable to deliver a quality product to its customer's negative word of mouth could prevent the company getting new customers and hence will not be able to survive in the modern competitive market. An equally acclaimed authority in software development Pressman claims that achieving quality products requires applying a high-quality process throughout development, integration, and testing. Software quality assurance is an umbrella activity that is applied throughout the software process and it consists of set of auditing and reporting functions that affects the effectiveness and completeness of quality control actions. (Pressman, 2010, pp. 413) This helps to make sure that any agreed upon standards and procedures are followed and most importantly to ensure that the problems are found and dealt with.

At early stages where the economy was not growing fast and the customer needs were not very complicated, the traditional development methodologies such as Waterfall, V shaped and Spiral had seen sufficient to cater to the user requirements. But as time progressed there was a requirement for complex systems in a shorter period. Hence in 2001 Kent Beck and 16 other software developers, writers and consultants signed the "Manifesto for Agile Development" (Pressman, 2010, pp.65) Agile development methodologies (such as XP, Scrum, and ASD) focus on higher customer satisfaction, lower defect rates, faster development times and a solution to rapidly changing requirements. Traditional approaches (such as Water fall, Spiral, or CMM-based methods) focused on predictability, stability, and high assurance. However, both approaches have situation dependent shortcomings that, if left unaddressed, could lead to project failures.

As discussed above information technology is undergoing continuous improvement. To keep pace, software development organizations need to release business-critical software in less time, but this venture often results in compromised quality. So the question arises: How can a company save time and reduce costs without sacrificing quality?

The aim of this research is to identify the reasons why companies are rapidly moving in to agility and whether they were able to achieve the expected software quality in agile development, compared to traditional process models. Finally, this research aims to explore how companies can achieve/strike a balance between software quality and the agility.

1.2 BACKGROUND

1.2.1. CONCEPTUAL BACKGROUND

Quality is the degree to which a system, component, or process meets customer or user needs or expectations.(IEEE Std. 610-12-1990) When considering the user expectations in relation to software products it is not only receiving a defect free product, but also receiving the product on time and within the expected budget, with relatively less defects. The Literature Review Chapter begins at page 11 of this thesis discusses further on quality and quality attributes.

Software Quality Assurance (SQA) defines and conducts the activities required to ensure software quality. (Pressman, 2010) According to IEEE standards SQA is "A planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements." (IEEE Std. 610.12-1990)

Due to the growing market conditions (Callen, 2007) today, software development companies need to incorporate requirement changes even in the later stage of development to cater to the frequently changing user requirements. But this should not result in longer periods to release a working product and should not exceed the budget and should be relatively defect free. (Pressman, 2010, pp.65)

Agile SW development combines a philosophy and set of development guidelines. The philosophy encourages customer satisfaction, early incremental delivery of software, highly motivated project teams, informal methods and overall development simplicity. The development guidelines stress delivery over analysis, design and active and continuous communication between developers and customers. (Pressman, 2010, pp.65) Whereas, the traditional models used in times where the requirements for a problem are well understood and not very complicated. There are number of software development process models that can be categorized as 'Prescriptive process

models' (such as Waterfall model, V-model), 'Incremental process models' and 'Evolutionary process models' (such as Spiral model) (Pressman, 2010, pp.61-62)

Irrespective of the process model is utilized, a generic process framework for software engineering defines five framework activities namely communication, planning, modeling, construction and deployment. (Pressman, 2010, pp.31) These five activities are common to any software engineering model. The process flow describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time (Pressman, 2010).

1.2.2 CONTEXTUAL BACKGROUND

Over the last ten years Sri Lankan IT industry has been growing exponentially and has eventually formed into the following three distinguishable segments (Saparamadu, n.d)

- Software Application Product Market
- Software Services
- Offshore Development

In the application product market the Sri Lankan IT companies have concentrated on software development for various industries and managed to position themselves in the global market.

The corporations with the world's leading IT companies gave Sri Lanka much exposure in the SW services field.

Offshore development is a type of offshore outsourcing where Sri Lankan SW companies have gained considerable amount of projects because of their ever growing talent. Currently over 50, 000 are employed in the IT and BPO industry in Colombo

and the work force is growing at over 20 % year on year. Sri Lanka boasts high levels of education with one of the highest literacy rates in South Asia (Saparamadu, n.d).

Sri Lanka is ranked among the top 50 Global Outsourcing destinations by 'AT Kearney' and ranked among top 20 Emerging Cities by 'Global Service Magazine'

With the above information we can come to a conclusion that Sri Lanka is very much suited for SW development and out sourcing activities and the country has a well educated and literate population. The investment climate is one of the best in the region.

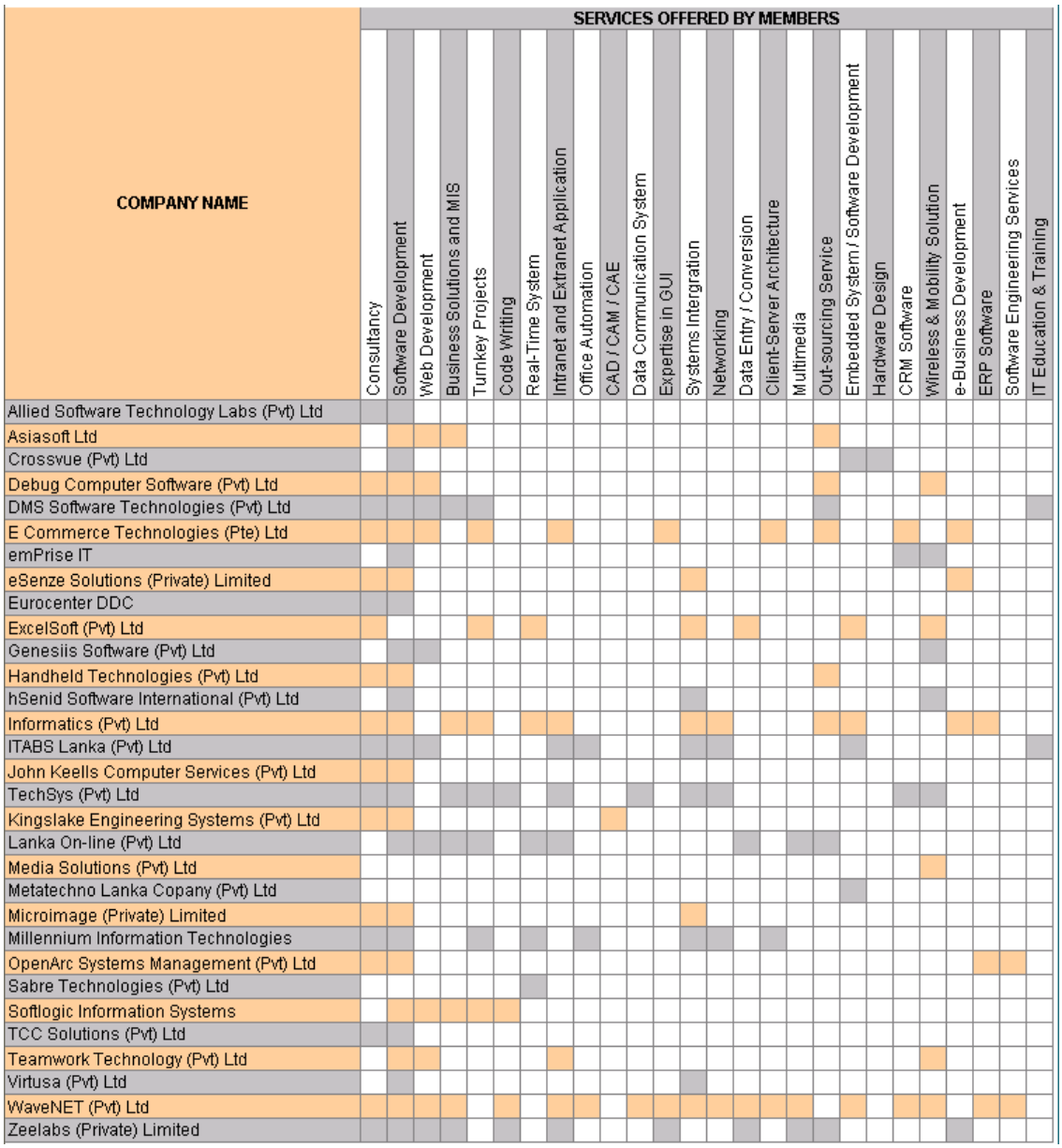
Due to the emerging demand in the IT industry in Sri Lanka there are number of ICT associations established in the country and few of them are listed below

- Sri Lanka Association for the Software Industry (SLASI)
- Sri Lanka Software Exports Association (SEA)
- Sri Lanka Association of Software & Service Companies (SLASSCOM)
- Information and Communication Technology Agency (ICTA)

Since this research is focused only on the companies registered with the SEA brief description of the association is stated below.

The Association, which was formed in 1999, had a membership of 39 ICT companies. Today the membership of the association has increased to 47 registered companies. The SEA focuses on developing Sri Lanka as an international software marketplace through exports of world-class software. Both the Board of Investment of Sri Lanka and Export Development Board are patrons of the Association. The SEA is also affiliated to the Ceylon Chamber of Commerce (Anon, Sri Lanka Export Development Board) Figure 16 summarizes the companies in relation to the Services offered, Industry served and the Technology focus.

(Source: Anon, Island software)



COMPANY NAME	SERVICES OFFERED BY MEMBERS																										
	Consultancy	Software Development	Web Development	Business Solutions and MIS	Turnkey Projects	Code Writing	Real-Time System	Intranet and Extranet Application	Office Automation	CAD / CAM / CAE	Data Communication System	Expertise in GUI	Systems Intergration	Networking	Data Entry / Conversion	Client-Server Architecture	Multimedia	Out-sourcing Service	Embedded System / Software Development	Hardware Design	CRM Software	Wireless & Mobility Solution	e-Business Development	ERP Software	Software Engineering Services	IT Education & Training
Allied Software Technology Labs (Pvt) Ltd	■	■		■		■		■		■		■		■		■											
Asiasoft Ltd		■	■	■														■									
Crossvue (Pvt) Ltd		■																	■	■							
Debug Computer Software (Pvt) Ltd	■	■	■															■				■					
DMS Software Technologies (Pvt) Ltd	■	■	■	■	■													■									■
E Commerce Technologies (Pte) Ltd	■	■	■		■			■				■				■		■			■		■				
emPrise IT		■																		■	■	■					
eSenze Solutions (Private) Limited	■	■											■										■				
Eurocenter DDC	■	■																									
ExcelSoft (Pvt) Ltd	■				■		■						■		■				■			■					
Genesiis Software (Pvt) Ltd		■	■																			■					
Handheld Technologies (Pvt) Ltd	■	■																■									
hSenid Software International (Pvt) Ltd		■											■									■					
Informatics (Pvt) Ltd	■	■		■	■		■	■					■	■				■	■				■	■			
ITABS Lanka (Pvt) Ltd	■	■	■					■	■				■	■					■								■
John Keells Computer Services (Pvt) Ltd	■	■											■	■													
TechSys (Pvt) Ltd	■	■		■	■	■		■			■		■	■							■	■					
Kingslake Engineering Systems (Pvt) Ltd	■	■								■																	
Lanka On-line (Pvt) Ltd		■	■	■	■		■	■							■		■	■									
Media Solutions (Pvt) Ltd																						■					
Metatechno Lanka Copany (Pvt) Ltd																			■								
Microimage (Private) Limited	■	■											■														
Millennium Information Technologies	■	■			■		■		■				■	■		■											
OpenArc Systems Management (Pvt) Ltd	■	■																						■	■		
Sabre Technologies (Pvt) Ltd							■																				
Softlogic Information Systems		■	■	■	■	■																					
TCC Solutions (Pvt) Ltd	■	■																									
Teamwork Technology (Pvt) Ltd		■	■					■														■					
Virtusa (Pvt) Ltd		■											■														
WaveNET (Pvt) Ltd	■	■	■	■		■		■	■		■	■	■	■	■	■	■		■		■	■		■	■		
Zeelabs (Private) Limited	■	■	■	■		■		■				■			■		■	■					■				

Figure 01: SEA Registered Company List

1.3 PROBLEM STATEMENT

In a modern economy it is often difficult to predict how a computer based system evolves as time passes. Market setting change quickly, end user needs evolve, and new competitive threats emerge without warning. In order to cater to this situation the

software development companies should be sufficiently capable to incorporate requirement changes even in a later stage of the development, but should deliver working products in a short time. Thus, most of the IT companies are moving into agile development these days to respond to this fluid business environment (Highsmith, 2001).

Once a software product undergoes the different stages of a development phases, a major task is to employ software quality assurance strategies to ensure the adaptation of the end product in the company's environment. The quality assurance of a software product is an ongoing process, which begins in/during the very early stages of the development (Daniel, 2004). The product designed by software Development Company must be process compliant to function successfully in a business environment.

Hence the burning question within the researcher of this thesis is 'Can the Software Development Companies in Sri Lanka achieve expected Software Quality through Agile Development?'

In the process of finding the answers for the above problem the very question this research tries to find the answer is 'is there a quality difference between the software products developed using Agile and Traditional methods and if there is a difference what is the healthier method?'

1.4 RESEARCH AIM & OBJECTIVES

A qualitative and quantitative study is carried out to identify the common software development models in the Sri Lankan IT industry. The study is mainly highlight whether the companies are able to achieve expected software quality through agile development. The specific objectives of this study can be stated as:

Aim – To determine the most suitable software development methodology in achieving high quality in software products

Objectives

- To identify the Software Quality factors.
- To identify the Traditional Software development models.
- To identify the software quality gap between Agile and identified traditional method for each of the identified quality factor.
- To identify the appropriate development method to attain each of the identified quality factor

1.5 SIGNIFICANCE OF THE STUDY

As already mentioned Information Technology is undergoing constant innovation. To keep pace, software development organizations need to release business-critical software in less time, but this increases risk often results in compromised quality. So the question arises: How can a company save time and reduce costs without sacrificing quality?

For a company to survive and to be a pioneer in any industry it has to expand the market, while retaining existing customers and attracting new customers. Moreover the SW development company needs to cater to its customers with quality products.

If the company's product is not up to the user's requirements it will not receive further projects from the existing client, as well as negative word of mouth will prevent the company getting new customers.

This research identifies whether the companies are able to achieve the quality aspects with agility compared to traditional models and how companies can balance between agility and software quality.

Such a study will help the companies to use most appropriate process model in their development. This, on the one hand, will lead the companies to develop SW products on time, within the budget and relatively error free, and on the other hand help companies to retain existing customers and attract new customers through positive word of mouth publicity. The research will bring industrial attention to the software quality in agile development and will guide the Sri Lankan SW development companies to face the challenge of global competition successfully.

1.6 RESEARCH LIMITATIONS

Due to the vastness of the Software Development sector, this study is confined to the Software Development organizations registered with the 'Software Exports Association' (SEA) in Sri Lanka. The research focuses only on the organization oriented quality factors. Based on the results of the pilot survey user oriented quality factors are excluded from this research (refer page 73 for details). Since it has found from the pilot study that the Waterfall model is the most common traditional method, the research compares the Agile method with the Waterfall model.

1.7 THESIS STRUCTURE

This thesis is structured in two parts. The first part includes the chapters two and three, focuses on the previous findings and the methodology used. Chapter 2 is dedicated to literature review of software quality assurance in Agile development. This review of the literature includes the basic quality related terminologies, popular quality models and quality management philosophies in its first three sections and further moves toward discussing the other related researches. Chapter 3 is a methodological chapter and describes the research methods, variables, data collection and data analysis. The second part of the thesis includes the chapters Four, Five and Six and it is focuses on data analysis and empirical results, Discussion of findings & Conclusions and finally the recommendations of the study.

CHAPTER TWO

2. LITERATUR REVIEW

2.1 INTRODUCTION

The chapter provides a comprehensive review of published work from secondary sources related to the topic of this thesis. The chapter helps in identifying the subject area precisely through what has been already proven. The Literature Review is presented in five sections as follows,

- Defining terminologies (Quality, Software Quality and Quality Assurance)
- Identifying software quality factors via various quality management philosophies and few famous quality models.
- Description of Software development process models and its evolution
- Agile Software development and Waterfall Software Development
- An analysis of other related researches

The first section of the chapter stats by defining the term quality and evolves the discussion through describing the quality management philosophies, popular quality models and software development process models respectively in the sections 2.2, 2.3, 2.4, 2.5 and lastly the section 2.6 discusses about the other related researches.

2.2 TERMINOLOGY

What is quality?

In the literature different people have defined quality in various different ways and there were no one specific definition stated for the term ‘Quality’. Also in the

literature quality is defined from various perspectives. Few definitions for the term quality from different perspectives are stated below;

- Customer Based
    - Quality can be defined as the degree to which a product, process or a service meets the requirements (Madura, 2007)
    - Quality is fitness for use (Juran J M, 1988)
    - Quality consist of the capacity to satisfy wants (Edwards C D, 1968)
- Manufacturing Based
    - Quality is the degree to which a specific product confirms to a design or pacification.(Gilmore H L, 1974)
    - Quality [means] conformance to requirements (Crosby P B, 1979).
- Product Based
    - Quality refers to amount of the unpriced attributes contained in each unit of the priced attribute (Leifler K B, 1982)
- Value Based
    - Quality is the degree of excellence at an acceptable price and the control of variability at an acceptable cost (Broh R A, 1982)

What is Quality Software?

Quality software is a software product which is reasonably defect-free, delivered on time and within the budget, meets requirements and/or expectations, and is maintainable. (Raman, 2009)

However, quality is obviously a subjective term. It will depend on who the 'customer' is and their overall influence in the scheme of things. For a software product the term ‘customer’ might include end-users, customer acceptance testers, customer contract officers, customer management, development organization's management/

accountants/ testers/salespeople, future software maintenance engineers, stockholders, magazine columnists, etc. Each one of them will have their own aspects on 'quality'. For example, an end user might define quality as friendly and defect free while the accounting department might define quality in terms of profit (Hoyer, 1996).

What is Software quality assurance?

The IEEE standard ANSI/IEEE 730-202 defines software quality assurance as a "Planned and systematic pattern of all actions necessary to provide adequate confident that an item or product confirms to established technical requirements". By going down the path of IEEE definition two major camps defines the SW quality. (Hoyer & Hoyer, n.d, pp.53-62)

- Conformance of the specification: quality defines in terms of the level which the product or service meets its written specification.
- Meeting customer needs: Satisfying customer explicit or implicit needs irrespective of any measurable product or service characteristics.

Currently software quality assurance is measured in two ways, from technical perspective and from the user perspective. (Kokol et al., 1991)

In the technical perspective of measuring SW quality is based on specifications. Developers measure quality and assure specification in terms of errors in code through testing process and through other mechanisms such as formal specifications and structured programming (Musa et al., 1990)

The end user perspective of software quality is measured through user experience to denote how well software meets user expectations. User dissatisfaction does not necessarily result from failure to meet specifications or coding errors, but can occur by delays in delivering the product, as well exceeding the estimated budget.

2.3 SOFTWARE QUALITY MANAGEMENT PHILOSOPHIES

This section presents different philosophies of quality from view points of quality management experts. These quality management philosophies proved a good alternative to formalize quality models on which the research is based. Quality management requires customer satisfaction, prefers prevention to inspection, and recognizes management responsibility for quality. (Schwallbe, 2004)

2.3.1 FOURTEEN POINTS FOR MANAGEMENT - DEMING

Walter Edward Deming defines quality in terms of customer satisfaction (Deming, 1988) Customer satisfaction is beyond conformance to specifications. According to Deming, the judge of quality should be the end user or the customer. Deming argues that the management system should be implemented in a way that everyone in the organization is responsible for quality of their output to the internal stake holders. He introduced fourteen points for management for people to understand and implement necessary quality transformation.

1. Create constancy of purpose for improvement of product and service. Stay in business and provide jobs through innovation, research, constant improvements and maintenance.
2. Adopt a new philosophy: For the new economic age, management needs to take leadership for change into learning organization.
3. Case dependence on mass inception: Eliminate the need for mass inception by building quality in to the product.
4. End awarding business on price: Aim at minimum total cost and move toward single suppliers.

5. Constant improvement of the system of production and service: Improvement is not a onetime effort. Management is obligated to continually look for ways to reduce waste and improve quality.
6. Institute training: Workers should train properly on their jobs and learn by objective method.
7. Institute leadership: Leading shall consist of helping people to do a better job and to lean by objective methods.
8. Drive out fear: To assure better quality and productivity, people should feel secure.
9. Break down barriers between departments: Team work culture across departments
10. Eliminate slogans, exhortations and numerical targets: Let workers formulate their own slogans. Then they will be committed to the contents.
11. Eliminate numerical quotas or work standards: Quotes take into account only numbers, not quality or methods. They are usually a guarantee of inefficiency and high cost including doing damage to the company.
12. Remove barriers to taking pride in to workmanship: People are eager to do a good job and distressed when they cannot achieve targets.
13. Institute a vigorous program of education: Both management and the work force should to be educated in the knowledge and understanding including team work and statistical techniques.

Take action to accomplish the transformation: It will require a special top management team with a plan of action to carry out the quality mission. (Deming, 1988)

A critical mass of people in the company must comprehend the fourteen points

2.3.2 THE IMPORTANCE OF THE TOP MANAGEMENT COMMITMENT TO QUALITY - JURAN

Joseph M. Juran proposes following two meanings to quality (Juran, 1988)

1. Quality consists of those product features which meet the needs of customers and thereby provides product satisfaction.
2. Quality consists of freedom from deficiencies

Juran's hand book proposes quality as "Fitness for use" rather than meeting "Customer satisfaction", (Juran J M, 1988) arguing that it is not a feasible task to meet customer needs. His view is much closer to the thought – "Conformance to Specification." The three elements of the Juran trilogy are as follows;

1. Quality planning: A process that identifies the customers, their requirements, the product and service features customers expect, and the processes that will deliver those products and services with the correct attributes and then facilitate the transfer of knowledge to the producing arm of the organization.
2. Quality control: A process in which a product is examined and evaluated against the original requirements expressed by the customer. Problems detected are then corrected.
3. Quality improvement: A process in which the sustaining mechanisms are put in place so that quality can be achieved on a continuous basis. This includes allocating resources, assigning people to pursue quality projects, raining those involved and pursuing projects and in general establishing a permanent structure to pursue quality and maintain the gains secured. (Juran, 1988)

2.3.3 STRIVING FOR ZERO DEFECTS - CROSBY

Philip B Crosby is a strong advocate of "Conformance to specification". Crosby summarizes his perspective on quality in fourteen steps built around following four fundamental "absolutes" of quality management.

1. Quality is defined as conformance to requirement; not as "goodness" or "elegance"
2. The system for causing quality is prevention, not appraisal. That is, the quality systems for suppliers attempting to meet customer's requirements are to do it right the first time. Crosby is a strong advocate of prevention, not inspection. In a Crosby oriented quality organization everyone has the responsibility for his/her own work. There is no one else to detect errors.
3. The performance standard must be zero defects, not which "close enough". Crosby has advocated the notation that zero errors can and should be the target.
4. The measurement of quality is the cost of quality. Cost of imperfection, if corrected, has an immediate beneficial effort on bottom line performance, as well as on customer relations. (Crosby, 1979)

2.3.4 FISHBONE DIAGRAMS - ISHIKAVA

Kauru Ishikava defines quality as "meeting customer needs". (Ishikava K, 1985) He further argues that no specific quality standard could ever design or meet the expected quality levels. According to Ishikava, quality is a very broad concept which goes beyond product, process, service, information quality etc... He introduced quality circles through Fish bone diagrams.

2.3.5 TOTAL QUALITY CONTROL - FEIGENBAUM

Armand Baum built his thought around a concept "Total quality control" (Baum, 1983). Baum states that quality is a dynamic which must be defined in terms of customer experience. He further states that quality should satisfy customer's explicit and implicit needs (Baum, 1983).

2.4 SOFTWARE QUALITY MODALS

The previous section of this chapter focused on different viewpoints of quality management gurus. These points could be helpful in solving common quality management problems in Sri Lankan Software Development companies. Quality management philosophies presented in the previous section represents flexible and qualitative view points of quality; this section presents rigid and qualitative quality structures which is a roadmap of identifying independent variables for current study.

2.4.1 MCCALL'S QUALITY MODEL

(Source: McCall, 1977, pp.4)

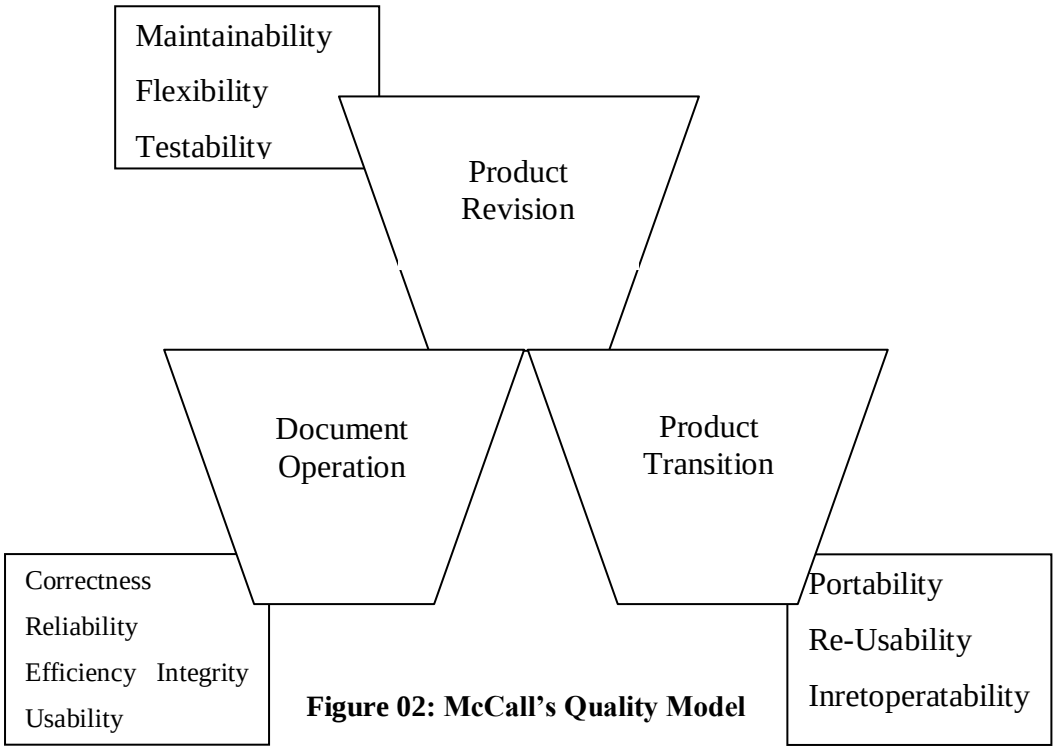


Figure 02: McCall's Quality Model

Jim McCall's quality model is primarily focused on system developers and the development process. However, he has tried to bridge the gap between users and developers by causing on number of quality factors, considering both users and developers priorities. (McCall, 1977, pp.3) The quality model illustrated in the diagram below is organized around three quality characteristics.

2.4.2 BOEHM'S QUALITY MODEL

(Source: Boehm, 1978, pp.25)

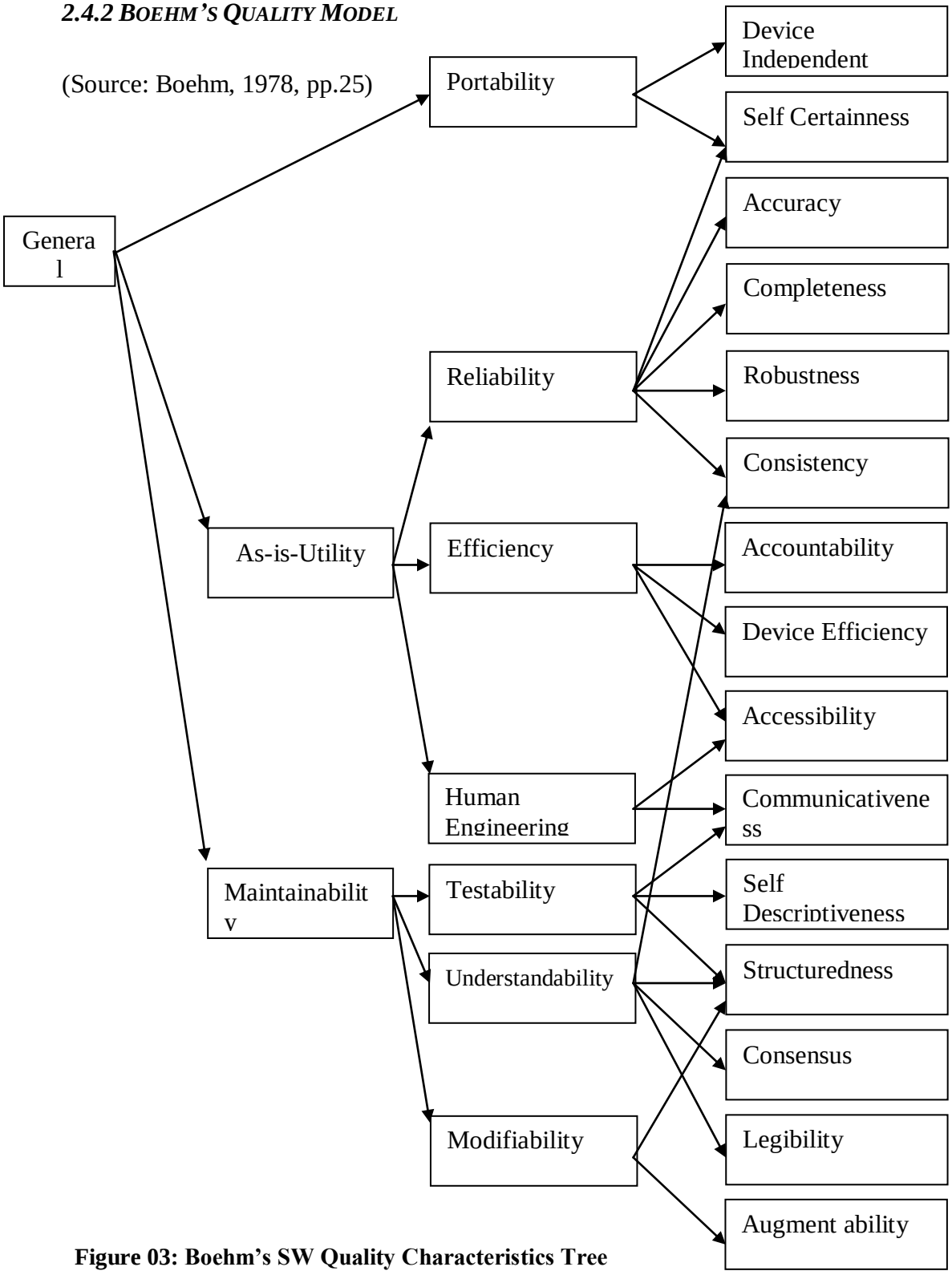


Figure 03: Boehm's SW Quality Characteristics Tree

Berry W Boehm's model illustrated in the above diagram has similarities to McCall's quality model. His qualitative approach defining quality stems from three levels in the hierarchy, which ends with primitive characteristics. (Boehm, 1978) These primitive characteristics individually contribute to overall quality level.

2.4.3 ISO 9126

Among the ISO 9000 series of quality standards, ISO has released the ISO 9126: Software Product Evaluation depicted in the diagram below. (ISO/IEC, 2001)

Source: Zhu, 2007, pp. 32

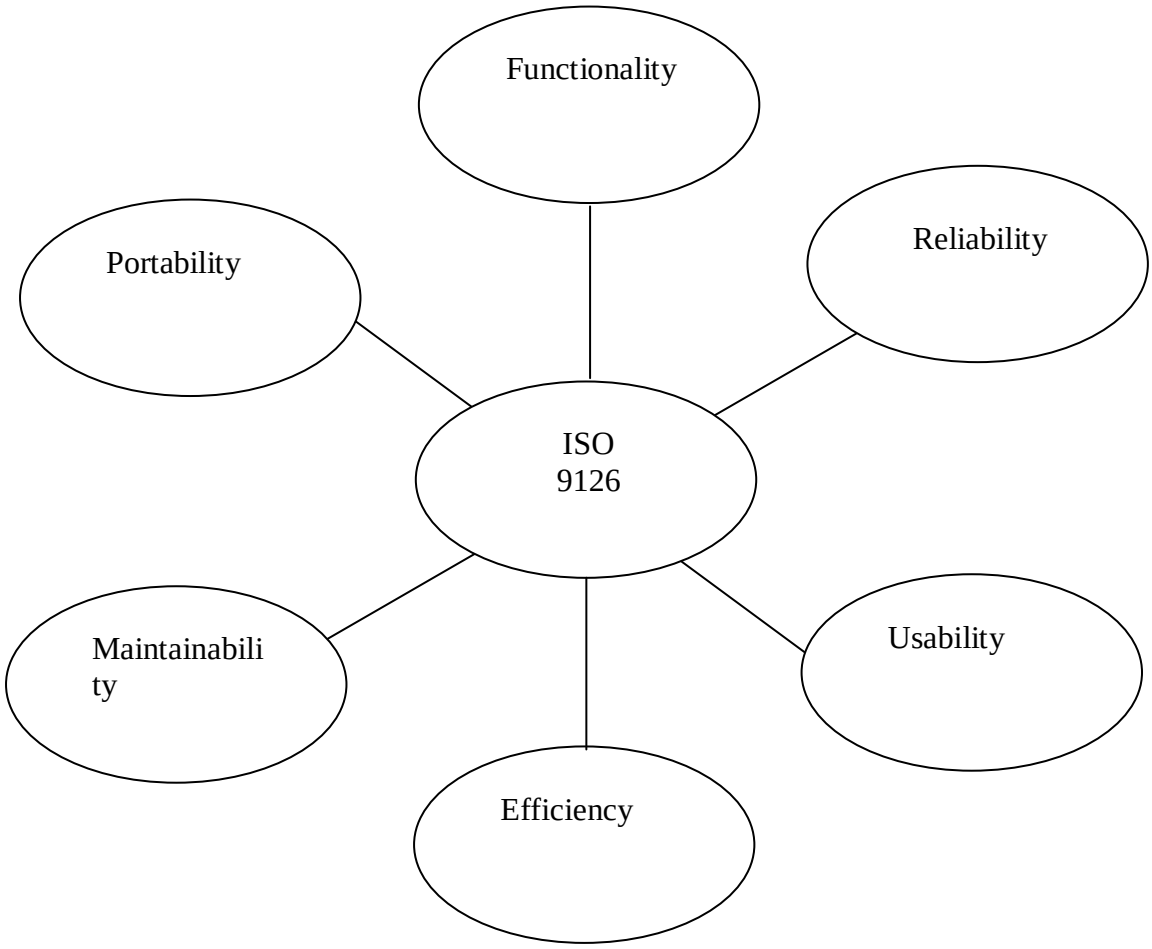


Figure 04: ISO Quality Model

ISO further propose quality characteristics/guidelines to evaluate the six areas of importance mentioned above.

(Source: Zhu, 2007, pp. 34)

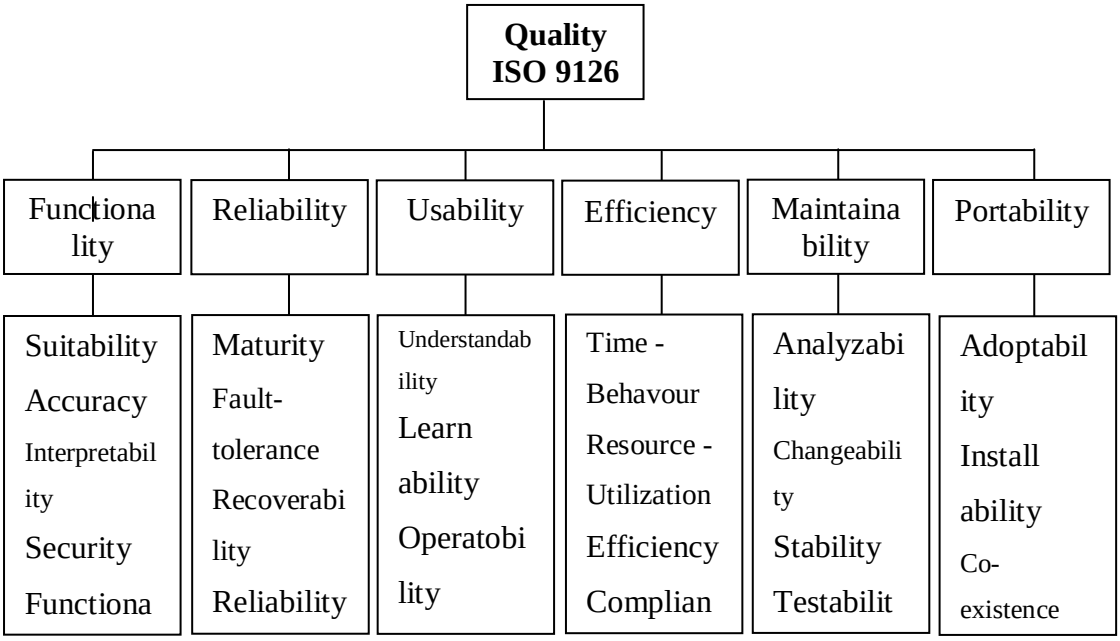


Figure 05: The ISO Quality Attributes

Each quality factor/ six areas of importance is represented by sub factors as depicted in the above diagram.

2.4.4 CMMI

The Carnegie Mellon Software Engineering Institute (SEI) introduced Capability Maturity Model Integration to evaluate organizations maturity levels through process management, project management, and engineering and support maturity levels. CMMI defines five stages based on maturity levels. (Emanuel et al., 2000)

2.4.5 FRUPS QUALITY MODEL

Robert Grady has presented similar model to McCall and Boehm. FRUPS stands for Functionality, Usability, Reliability, Performance and Supportability. FRUPS categories have been divided into two main categories; Functional and Non-

Functional. (Grady, 1992) These categories are usable in assessing product and requirement quality levels.

Considering the above SQA philosophies and the various quality models it has identified that the most of the quality factors in different models overlap with each other. And it appears that almost all the models and philosophies have mainly focused on the Users Requirements, Testability, Maintainability and Portability. When considering the User requirements it includes not only adhering to their functional requirements, but also delivering the product on time and within the expected budget.

When summarizing the literature found above, software quality can be categorize as:

- Project quality - deliver the product on time within the budget
- Process quality - effectiveness, , predictability, repeatability, improvement (Tyrell,2000)
- Product quality - reliability, correctness, durability, maintainability , testability, installability ect

2.5 SOFTWARE ENGINEERING PROCESS MODELS

This segment of the literature survey presents different software development process models commonly used in the industry. These process models help the researcher to understand the software engineering process and different process flows and the evolution of the software development models. The section starts by defining "Software Process" and evolve through identifying generic frame work present in every SW engineering process, irrespective of the model. Furthermore, the section describes the evolution with some conventional models, their strengths and weaknesses and move towards explaining the Agile software development and the importance of the agility in the modern SW Engineering work.

2.5.1 SOFTWARE PROCESS & GENERIC PROCESS MODEL

Pressman commenting on the SW process says thus; “When you work to build a system, it is important to go through a series of predictable steps – a road map that helps you create a timely high quality result. The road map that you follow is called a ‘software process’” (Pressman, 2010, pp.31). A process can be defined as a collection of work activities, actions and tasks that are performed when some work product is to be carried. (Pressman, 2010, pp.15)

A process frame work provides the basis for a complete SW Engineering cycle by discovering few frame work activities that are applicable to any SW project, irrespective of size and complexity. Pressman had described five generic activities encompassed in the SW engineering processes as follows.

Communication – Before starting any technical work it is extremely important to communicate and collaborate with the customer about his/her requirement.

Planning – This helps to simplify the work to be done. A SW project is a complicated journey, and planning creates a ‘map’ that helps to guide the team

Modeling – Prior to the construction of the system, model it for better understand the SW requirements and design the system to achieve those requirements.

Construction – This consist of the activities related to code generation and testing to uncover errors in the code.

Deployment – Delivery of the software to its end user. This can be done as a whole or partially completed increment so that the customer can evaluate the product and provide feedback to the supplier. (Pressman, 2010, pp.39)

2.5.2 EVOLUTION OF SOFTWARE DEVELOPMENT

The primary function of software development process models is to determine the order of the stages involved in software development and to establish the transition criteria for progressing from one stage to the next. (Boehm, 1988 pp. 61) During the history of software development, different models and approaches have been suggested for deal with the complexity and uncertainty of software development. Figure 2.5 shows the evolution of process models in the past decades. As can be seen in the y-axis of Figure 1, it has been suggested that the evolution of software development models originates from the problems of ad hoc programming that, at first, led towards traditional plan-driven models and towards iterative change-driven models of software development. (Basili & Reiter, 1981) The term 'ad hoc' is used to refer to the low degree of methodological discipline. It should also be noted that the positioning of the different software development models on the y-axis in Figure: 5 is illustrative rather than scientific.

(Source: Basili & Reiter, 1981, pp.45)

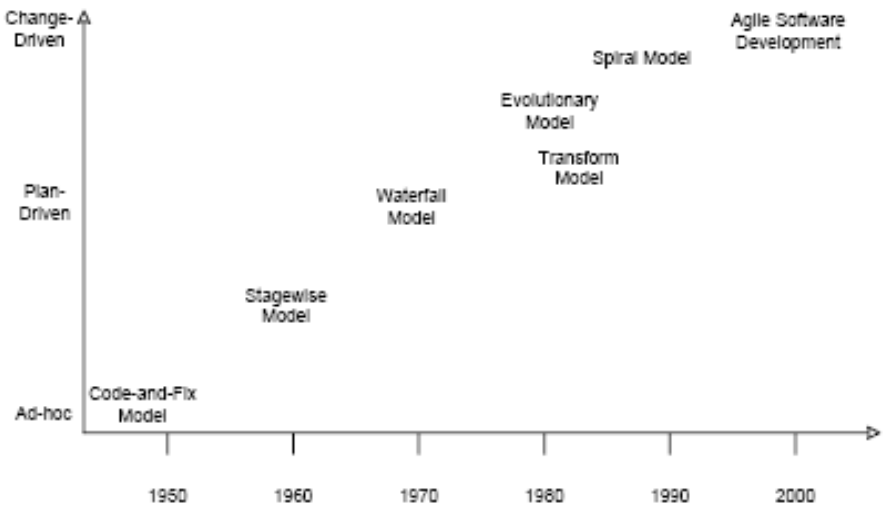


Figure 06: The Evolution of Software Process Models

2.5.2.1 Plan-Driven Models for Software Development

The plan-driven approaches of software development have been defined as document-driven, code-driven, and traditional process models (Boehm, 1988). As the names suggest, a common feature for the plan-driven process models is their emphasis on defining the scope, schedule, and costs of the project upfront including, for example, an early fixing stage and extensive documentation of the end product requirements. One common characteristic could also be the recurrence of the software development phases only once during the development process, i.e., with only hints of iteratively (Larman & Basili, 2003). In the following sections of this thesis, the process models of this category will be referred to as traditional software development.

The two-step process model of code-and-fix, used in the early days of software development, resulted in difficulties that necessitated explicit sequencing of the phases of software development (Boehm, 1988). In particular, the need to design prior to coding, to define requirements prior to design, and the need for early preparation for testing and modification were identified (Boehm, 1988). One of the first models to rise to that challenge was the stage wise model as early as in the middle of the 1950s (Benington, 1983). This model evolved from the problems caused by the increasing size of software programs, which could not be handled by a single programmer (Benington, 1983). In 1968, the NATO Science Committee held a software engineering conference in Garmisch, Germany, where the ‘software crisis’ or ‘software gap’, was discussed (NATO Science Committee 1969). A standardization of the software development process with an emphasis on quality, costs, and development practices was the key recommendation of the conference (Lycett et al., 2003). Soon after this, as a refinement of the stepwise model, the ‘waterfall model’ was introduced.

The Waterfall Model: Early version of the waterfall model was introduced in 1970 by Royce (1970) and it has since evolved into a concept consisting of the sequential phases of requirements analysis, design, and development (Larman & Basili, 2003). According to Boehm (1988), the waterfall model provided two main advances over the stepwise model: it introduced prototyping to parallel the stages of requirements analysis and design, and provided feedback loops between the sequential stages. It should also be noted that, already in the early waterfall model of Royce (1970), it had been realized that it might be necessary to first build a pilot model of the system, i.e., to conduct two cycles of development and to obtain feedback to adjust the model. Thus, hints of iterations in the model can be seen. This iterative feedback-based step has been lost in most descriptions of this model, although it is clearly not classic IID. (Larman & Basili, 2003, p.48) Today, the waterfall model has been adopted for most software acquisition standards in government and industry (Boehm, 1988). Though the waterfall model has solved various core problems in software development, it also includes features not appropriate for every software development context (Boehm, 1988). A central problem of the waterfall model has been identified as its emphasis on fully elaborated documents as completion criteria for early requirements and design phases. (Boehm, 1988, p.63) In the water fall model, a reasonably sequential approach is used. (Has a linear process flow) The result of one phase will virtually lead to the next. Phases need to be worked completely for the project to advance. This model is suitable at times where the requirements are fixed, stable and well understood. The main disadvantages are that this is not applicable to large and ongoing projects where the requirements are ever changing. And there is lack of quality assurance during phases. (Pressman, 2010, p.40)

(Source: Pressman, 2010, pp.42)

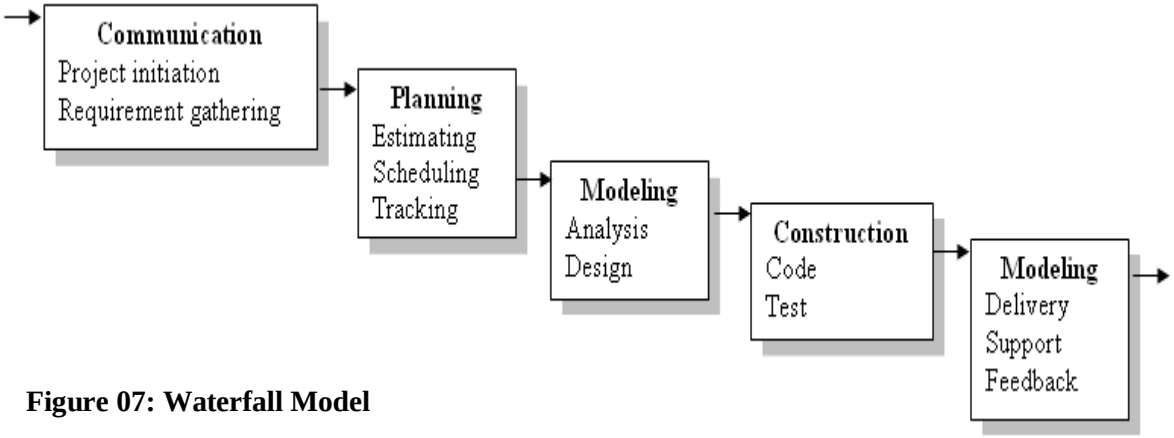


Figure 07: Waterfall Model

The V-Model: This model can be considered a variation of the waterfall model. The original V-Model includes similar phases to the waterfall model, but its phases are not defined as a linear activity but form a V-shape. The V-Model first became a standard for German civil and military federal agencies in 1997, as a result of the Development Standards for IT Systems of the Federal Republic of Germany. In this model, the Coding- phase is situated in the intersection of the V, while the software design, software verification, system design, system verification, and requirements engineering, system validation form the crescent counterparts each side of the V-shape. The model emphasizes traceability between the requirements, design and implementation. (Schauble, 2007) The lack of attention to quality assurance in the waterfall model was one of the reasons that have led to development of the V – model. As the team moves down the left base of the V, basic problem requirements gradually converts to more detailed technical representation. Once the code generation is done the team moves up the right side of the V where there is chain of tests (SQA activities) associated to validate each of the models created in the left side. Since the model is almost similar to the water fall model the same disadvantages are applicable to the V – model except lack of SQA activities. (Pressman, 2010)

(Source: Pressman PS. 2010, p.47)

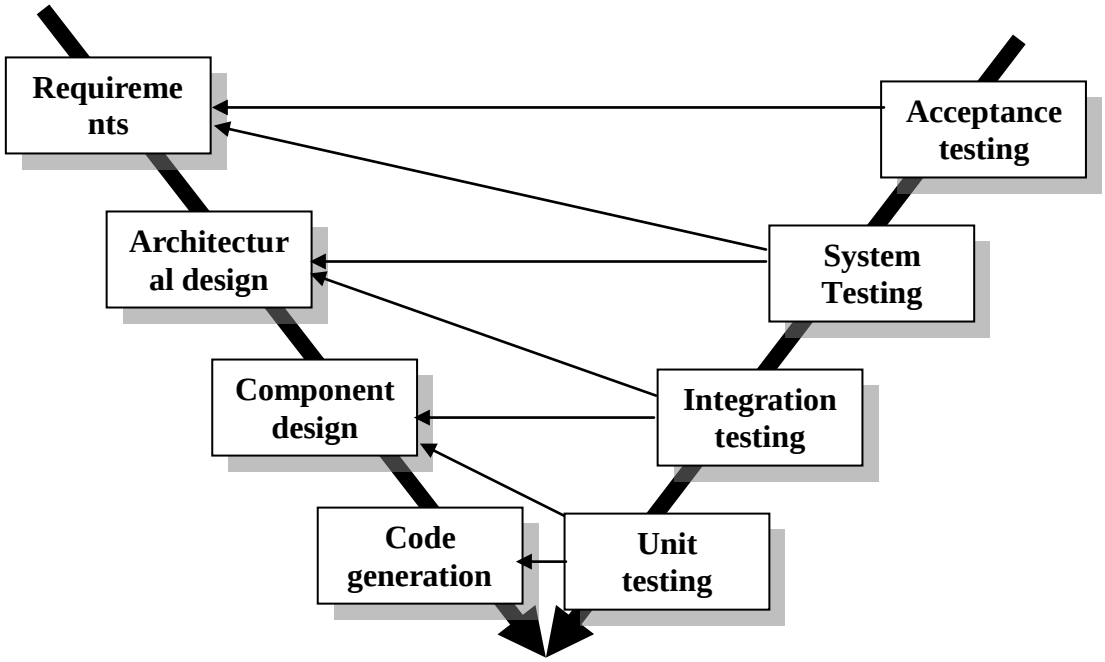


Figure 08: V Model

More commonly, it has been argued, that no life-cycle scheme, even with variations, can be applied to all system development. (McCracken & Jackson, 1982, p.30) On the other hand, according to the survey study of Fitzgerald and Wynn (2004), despite numerous existing software development methodologies, as much as 60% of software development organizations do not apply any development methodology. An additional problem has been identified in using a disciplined approach to software development; rather than focusing on the end (the development of software), developers become pre-occupied with the means (the software development method). (Fitzgerald & Wynn, 2004, p.65) In practice, the result may be the disparity between the organizational software development process and its actual implementation in the software development teams (Fitzgerald & Wynn, 2004).

Another dilemma identified among plan-driven approaches to software development, is the pursuit of certainty. The up-front requirements definition, and locking of the project scope, leads to contracts and decisions based on estimations of costs, time and resources. However, such estimates have been found to be highly prone to uncertainty

(Morien, 2005). Nonetheless, the success of software projects is often measured against these estimates as it may be appealing, from the viewpoint of both customer and supplier, to agree fixed costs, scope and schedule for the project up-front. However, it has been stated that certainty is a myth and is the most uncertain part of any project. (Morien 2005, p.519) In fact, it could be argued that the quest for certainty, in both time and money, may not only fail to pay off in these respects but may seriously affect the quality of the end product as well.

Hence, it can be argued that the plan-driven models of software development can and should be applied in a dynamic way by repeating the phases or even the entire process, if necessary. However, the original purpose of these process models was not to welcome changes during the development, but rather to try to fix factors, such as scope, time and money, up-front in order to eliminate change which was considered a risk factor.

2.5.2.2 Iterative Change-Driven Models for Software Development

The central software development models, developed after the waterfall model, seem to have the common aim of enabling, at least to some degree, the evolution of product requirements during the process of software development. This contributed one main modification to the earlier software development models: the adoption of the iterative and incremental approach. Iterative development refers to the overall lifecycle model in which the software is built in several iterations in sequence (Larman, 1998). According to Larman (1998), iteration can be considered as a mini-project in which the activities of requirements analysis, design, implementation and testing are conducted to produce a subset of the final system, often resulting in internal iteration release. An iteration release has been defined as a stable, integrated and tested

partially complete system. (Larman, 1998, p.10) Incremental development involves adding functionality to a system over several releases, i.e., a repeated delivery of a system into the market or production. Thus, one incremental delivery may be composed of several iterations. A development approach where the system is developed in, several iterations is called iterative and incremental development (IID), yet it is often referred to as iterative development. (Larman & Basili, 2003) Even though agile software development has recently brought the IID approach of developing software into the spotlight, the history of these approaches is, in fact, considerably longer (Larman & Basili, 2003). Many of the earlier change driven approaches have adopted the ideologies of prototyping, for example, where the first early prototype gradually evolves into the final software product with no formal specifications or co-operation with the customer (McCracken & Jackson, 1982). Among the first models that focused on increasing the possibility of determining product improvements throughout the development process, was the evolutionary development model. This concept was first introduced in 1981 (Gilb, 1981) and has been expanded by Gilb (1988, 2005). This method suggested an iterative development approach in which the product increment was understood as a delivery to the real customer rather than a prototype (Gilb, 1981). While evolutionary delivery also lacks plans for future deliveries, it does attempt to capture feedback to guide future deliveries. This is in contrast to pure incremental delivery where the plan is drafted for several future deliveries and feedback is not the sole driving force (Larman, 1998).

The evolutionary model was followed by the transform model (Balzer et al., 1983), which is also based on the iterative development model and on adjusting the product during the development. The transform model, however, had a strong emphasis on product specifications due to its ideology of focusing on automatic transformation of specifications into code (Boehm, 1988). This approach had its origin in the problems

of the earlier software development models producing spaghetti code, which was difficult to modify and maintain (Boehm, 1988).

The Spiral Model: This was introduced in the late 1980s. The model typically consists of four iteratively repeatable steps as mentioned below:

1) Determining the objectives, alternatives, and constraints,

2) Evaluating alternatives, and identifying and resolving risks,

3) Development and verification, and

4) Planning the next phase. (Boehm, 1988)

Boehm (1988) defined the spiral model as a risk-driven approach for software development. In the spiral model, the iteratively evaluated strategy for resolving the risks of the next spiral has an effect on the choice of the software development approaches to be adopted. Depending on the risks, the spiral model then allows the adoption of any mixture of development approached, such as prototyping or elements from the specification-oriented waterfall approach modified to incremental development. According to Boehm, the risk-driven approach also means that the results of each risk analysis activity has an effect on the amount of time and effort allocated to the different development activities in the following spiral, while also influencing the required level of completeness, formality, or granularity of product specifications. (Boehm, 1988)

(Source: Pressman, 2010, p.47)

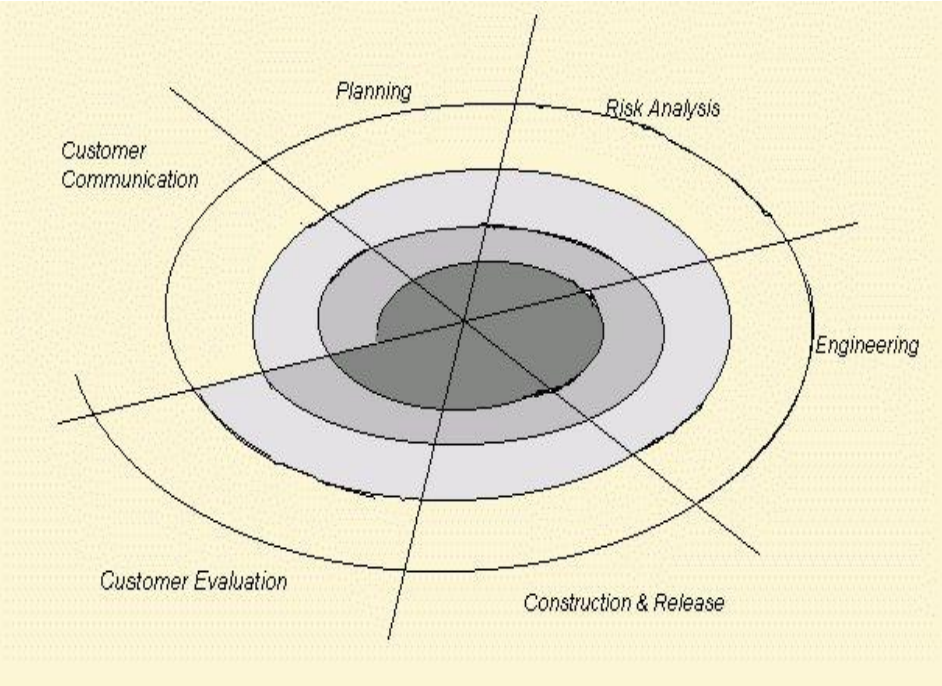


Figure 09: Spiral Model

The Unified Process - This is another iterative and change driven model that emerged in the early 1990s. This process model consists of use case-driven and risk-driven procedure. It also takes account of evolutionary software development based on iterations and increments. A unified process is an iterative and incremental software development process framework providing evolutionary feel that is essential in the modern software development. (Kroll et al, 2003) This model was introduced by James Rumbaugh, Grady Booch and Ivar Jacobson during the early 1990s and it is an attempt to draw on the best features and characteristics of traditional software process models. (Pressman 2010, pp.81)

Unified process divides the project in to 4 phases – Inception, Elaboration, Construction and Transition and the five generic frame work activities (communication, Planning, Modeling, Construction and Deployment) are included with no exception.

(Source: Lewis, 2006, p.46)

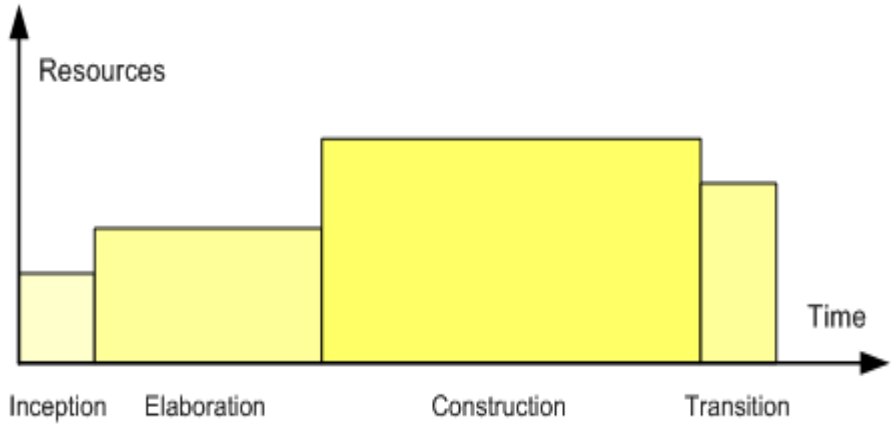


Figure 10: Phases in the Unified Process

The figure below depicts the relationship between UP phases and the general SW development activities.

(Source: Pressman, 2010, p.51)

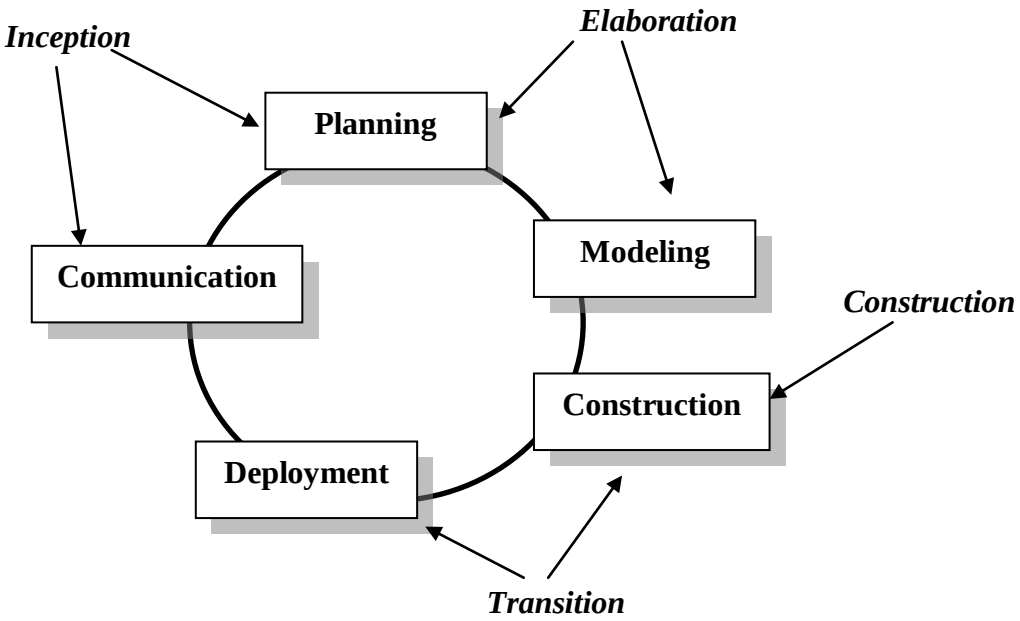


Figure 1: SWD Phases & Generic Activities

- **Inception**

  As shown in the diagram (Figure 9) this is the smallest phase in the project, and ideally it should be quite short. If the Inception Phase is long, then it is usually an indication of excessive up-front specification, which is contrary to the spirit of the Unified Process.

  The goals of the Inception phase can be list as follows:

  - Establish a justification or business case for the project
  - Establish the project scope and boundary conditions
  - Outline the use cases and key requirements that will drive the design tradeoffs
  - Outline one or more candidate architectures
  - Identify risks
  - Prepare a preliminary project schedule and cost estimate (Kroll et al,2003)

- **Elaboration**

  The Elaboration phase focuses on capturing system requirements as much as possible. The primary goals of Elaboration can be listed as follows:

  - To address known risk factors
  - To establish and validate the system architecture

  Activities carried out in this phase include the creation of use case diagrams, conceptual diagrams (class diagrams with only basic notation) and package diagrams (architectural diagrams).

  In order to achieve the second objective a partial implementation of the system is carried out; this includes developing core, most architecturally significant,

components. It is built in a series of small, time boxed iterations. By the end of the Elaboration phase the system should have stabilized executable architecture baseline that support the key system functionality and exhibit the right behavior in terms of performance, scalability and cost.

Elaboration phase's final deliverable is a plan that includes cost and schedule estimates for the Construction phase. Since this plan is based on the Elaboration phase experience should be accurate and credible.

The Lifecycle Architecture Milestone marks the end of the Elaboration phase. (Kroll & Kruchten P, 2003)

- **Construction**

  This is the largest phase in the project consisting of much iteration. In this phase the rest of the system is built on the foundation established in Elaboration phase. The features of the system are implemented in a series of short, time boxed iterations and each result in an executable release of the software.

  The Initial Operational Capability Milestone marks the end of the Construction phase.

- **Transition**

  The final phase of the project is Transition. A system that has been developed through all the above phases is deployed to its end user at this phase. Feedback received from an initial release (or initial releases) may result in further refinements to be incorporated in several Transition phase iterations. The Transition phase also includes system conversions and user training. The Product Release Milestone marks the end of the Transition phase.

Unified process can be described in two models namely Rational Unified Process and Agile Unified Process.

- Rational Unified Process
- Agile Process

Since the research is based on Agile development the literature survey section does not provide details of the Rational Unified Process (RUP). However, for the completeness of this section a pictorial representation of RUP is attached below (Figure: 11).

(Source: Witmann, 2005)

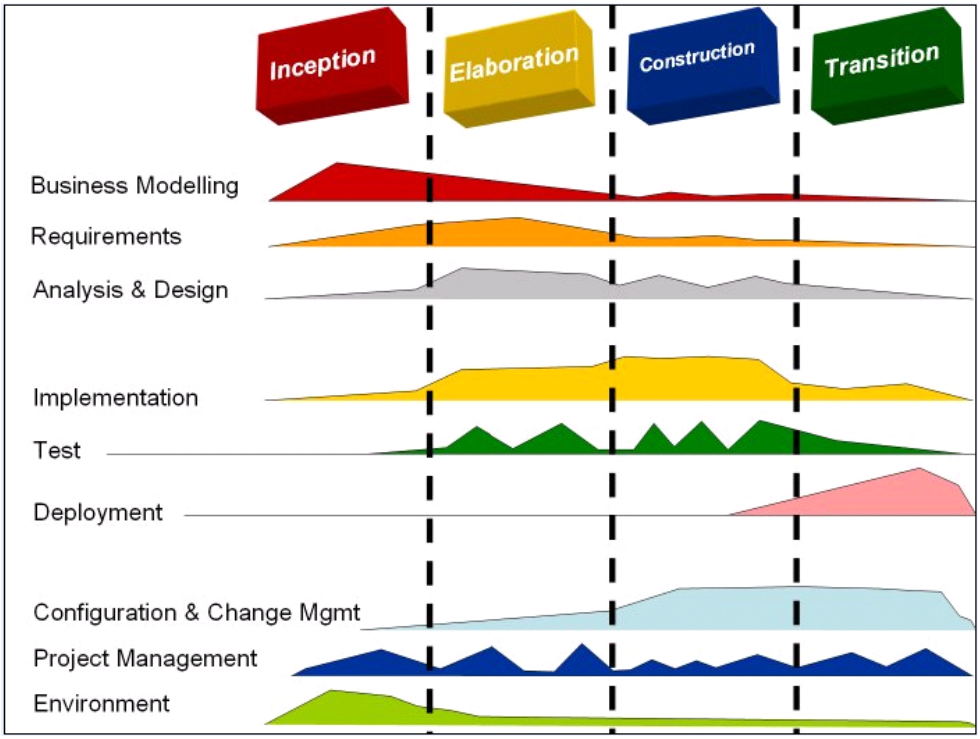


Figure 2: RUP Model

Agile software development, which emerged in the mid-1990s, can also be classified as an iterative and change-driven software development approach. It could be argued that at present there is no common agile process model with specified phases, but there is rather a set of fundamentals (Agile Alliance 2001)

The diagram below illustrates how Beck (1999) has compared the traditional process with the iterative process.

(Source: Beck, 1999, p.189)

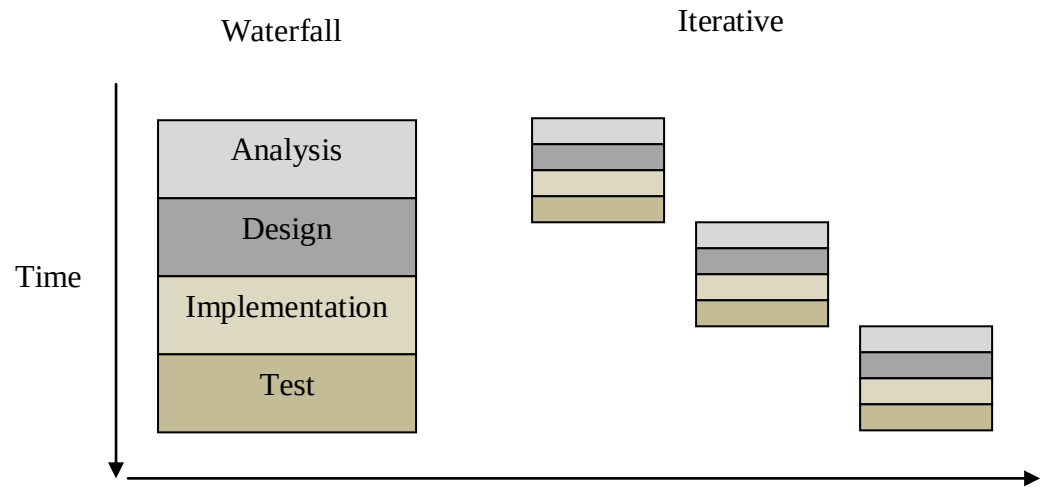


Figure 3: Model Comparison

According to Lorman "in modern iterative methods, the recommended length of one iteration is between one and six weeks", (Lorman, 2004 p.11), where as the "incremental deliveries are often between three and twelve months" (Lorman, 2004, p. 20). The principles of agile development suggests a short (i.e. From two weeks to two months) duration of development iterations. Evo also promotes relatively short delivery cycles of few weeks (Lorman, 2004). Similarly as in the evolutionary model agile methods also consider the term "iterative" as referring to evolutionary advancement of the product rather than just rework. (Lorman & Basili, 2003) The next section discusses the agile development in detail.

2.6 AGILE SOFTWARE DEVELOPMENT & WATERFALL SOFTWARE DEVELOPMENT

This section detailed both Agile and Waterfall methods including their history, fundamentals and current status. A few agile techniques common in the Sri Lankan context has also been detailed in the later part of the section.

2.6.1 WATERFALL SOFTWARE DEVELOPMENT

History of Waterfall Development

As the history says, originally waterfall model was mentioned by Dr. Winston W. Royce in 1970 in the article "Managing the Development of Large Software Systems: Concepts and Techniques". In the article he proposed what is now popularly referred to as the waterfall model as an initial concept. His paper then explored how the initial model could be developed into an iterative model, with feedback from each phase influencing previous phases, similar to many methods used widely and highly regarded by many today (Royce, 1970).

Phases of the Waterfall Model

The Waterfall Method is comprised of a series of very definite phases, each one run intended to be started sequentially only after the last has been completed, with one or more tangible deliverables produced at the end of each phase. According to the article published by Paul Smith Waterfall model can be classified in as 10 phase and 6 phase model. According to the article the 10 phase model consist of the following phases and respective deliverables. (Smith, 2011)

1. *Initiation Phase:* An opportunity is spotted, and is proposed in a formal Concept Proposal Document.

2. *System Concept Phase:*
    - Deliverables:
        - System Boundary Document (to define the scope or boundary of the concept),
        - Cost Benefit Analysis,
        - Risk Management Plan,
        - Feasibility Study. Typically evaluated in three areas: economical, operational, technical.

3. *Planning Phase:*
    - Used as a reference to keep the project on track and to evaluate the progress of the MIS team.
    - Provides the basis of acquiring the resources needed to achieve a solution.
    - Deliverables:
        - A Project Management Plan is developed.

4. *Requirements Analysis Phase:*
    - Deliverables:
        - Software requirement specification
        - Dataflow diagrams

5. *Design Phase:* The requirements are analyzed in order to design the product's architecture.
    - Deliverables:
        - Design of the Output Requirement including frequency, distribution, volume and format.
        - Design of the input layouts

- Finalized file design including File Name, Field Names, Field Type, Field Size, Primary Key and Foreign Keys.

6. *Development Phase:* The design is converted into reality and then white box tested by the development team.

7. *Integration and Test Phase:* The product is tested by the development team, Quality Assurance staff, and final users.

8. *Implementation/Deployment Phase:* The product is rolled out into a production environment.

9. *Operation and Maintenance Phase:* The system is monitored to ensure it continues to meet performance requirements, with periodic *In-Process Reviews* to suggest ways on improving the system.

10. *Disposition Phase:* The product is removed from service, with special emphasis on archiving the data, or moving to another system.

Six phase model for small organizations has been described in the article as follows (Smith, 2011)

- Initiation/Planning/Concept *Phase* (5% of the project)
- Requirements Analysis Phase (10% of the project)
- Design Phase (15% of the project)
- Development Phase (40% of the project)
- Integration and Test Phase (20% of the project)
- Implementation/Deployment Phase (10% of the project)

In Royce's original waterfall model, the following phases are followed in order; (Royce, 1970).

1. *Requirements specification*

SRS is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements.

2. *Design*

Software design is a process of problem solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

3. *Construction or coding*

Construction is the realization of a technical specification or algorithm as a program

4. *Integration*

Here the various codes designed by different programmers are integrated together

5. *Testing and debugging*

Methodical process of uncovering and reducing the number defects/bugs in the software product

6. *Installation*

Locate the program onto a computer system so that it can be executed.

Maintenance

Is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

Advantages of the Waterfall Model

As been described by Bhakti Satalkar the most important advantage of the model is that it imposes control, since the start and the end of each of the phases is well decided. This also helps in recognize the progress in the system, not only for the vendor, but also for the client. Because the requirements of the system along with the design are written down before hand, it guarantees that there is no waste of time or efforts. This in turn ensures that the system does not slip on the schedule. Writing the specification of the system in advance also ensures that the customer expectations are met. The written document helps the next team in the next phase, as all the details about the system are well mentioned in the document (Satalkar, 2011).

When the requirements and design are made before the start of the actual development of the system, the quality of the system is better. It also proves to be of help in identifying the flaws in the system and correcting them in advance. Due to clear demarcation of phases, knowledge transfer between the different teams is efficient (Satalkar, 2011).

Since the system is planned well in advance, the number of resources required to develop the system are also not many. There is clear distribution of work, which can be carried out as all the tasks are well defined in advance (Satalkar, 2011).

At the end of each phase, there is quality control and quality assurance activities been carried out. When the tests are carried out at the end of each of the phase, it helps in getting rid of bugs in the system, before the bugs give rise to some more bugs in the system. This in turn helps in quality control and quality assurance activities (Satalkar, 2011).

Disadvantages of the Waterfall Model

The very assumption that the all the system requirements can be frozen before the system is designed is the biggest disadvantage of the waterfall model. When a new system has to be designed more often than not the user of the system is not able to give all the requirements at one go and the requirement changing again is not new. If an existing system is to be automated, then this disadvantage no more remains a disadvantage (Satalkar, 2011).

Since the hardware and the software requirements are also frozen at the beginning of the project, the hardware and software chosen often becomes obsolete, as the software projects often taken long period of time to be completed. Another disadvantage of the system is that a working model is not available till the last stage of software development. Therefore the client is not able to find out any mistakes in the software, till the final version is given to him (Satalkar, 2011).

Another disadvantage of this software development model lies in its biggest advantage. One cannot go back to the earlier stage, once the development work has moved to the next phase. Therefore, in case there is a problem in the design phase, then the implementation phase and the further phases face a lot of problems. Due to this disadvantage was the modified waterfall model introduced, where one can go back to the previous stage in a loop (Satalkar, 2011).

2.6.2 AGILE SOFTWARE DEVELOPMENT

History and Fundamentals of Agile Development

The emergence of agile methodologies began in the mid 1990's, with the surfacing of Extreme Programming (XP) (Beck 1999), Scrum (Schwaber, 1995), eXtream Testing (Jeffries 1999), Crystal Family of Methodologies (Cockburn 1998), Dynamic System

Development Method (DSDM) (Stapleton 2003), Adaptive Software Development (ASD) (Highsmith, 2000) and Feature-Driven Development (FDD) (Code et al., 1999)

The principles of agile development can be traced back to lean manufacturing in 1940s, and Agile manufacturing in the early 1990s. Lean manufacturing is based on the fundamentals of short cycle time reduced setup, multi-skilling and flow being in place while driving out waste in time, activity, inventory and space (Ross & Francis, 2003). The essence of the agile approach in manufacturing has been summarized as "the ability of an enterprise to thrive in an environment of rapid and unprintable change" (Gould 1997, p28). While the debate between the actual differences of lean and agile is still continuing n the manufacturing sector (James 2005), the central ideologies of both can be found in the fundamentals and methodologies of agile software development. For example, in Lean software development (Poppendieck & Poppendieck 2003) the lean principles are integrated with agile practices.

In software development, the agile 'movement' was launched in 2001 when the various originators and practitioners of these methodologies met to identify the common aspects of these methods that both combined old and new ideas and clearly shared some particular ideologies in common. As a result, the manifesto for Agile Software Development was drafted and the term "agile" was chosen to combine the methods and techniques that would share the values and principles of the Agile Manifesto (Agile Alliance 2001) set out the central elements of agility that should be embedded in any method claiming to be agile. The agile manifesto emphasizes the agile values listed below on the left, while the items listed below on the right are still considered valuable.

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

As per the Agile Alliance the twelve principles of agile software development are mentioned below (Agile Alliance, 2001);

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. The welcoming of changing requirements, even late in development, for the benefit of the customer's competitive advantage
3. Frequent delivery of working software, the release cycle ranging from a couple of weeks to a couple of months, with a preference for a shorter timescale
4. Daily collaboration of business people and developers throughout the project
5. Building of projects around motivated individuals by offering them an appropriate environment and the support they need, and trusting them to get the job done
6. Emphasis on face-to-face conversation for conveying information and within a development team
7. Working software is the primary measure of progress
8. Agile processes promote a sustainable development pace for the sponsors, developers, and users
9. Continuous attention to technical excellence and good design enhances agility

10. Simplicity is essential for maximizing the amount of work not having to be done
11. Self-organizing teams give best results in terms of architectures, requirements, and designs
12. Regular reflection of teams on how to become more effective, and tuning and adjusting its behaviour accordingly.

The principles of agile software development can be considered as fundamental ideologies that should be embedded in the practices of any software development method claiming to be agile.

The core features of agility that should be embedded in any true agile method have been further specified as follows:

- Iterative development of several cycles,
- Incremental development,
- Enable the teams to self-organize and determine the management of work,
- Emergence of processes, principles, and work structures during the project.

(Boehm & Turner, 2003)

In addition, the active involvement of users in requirements and planning, and the importance of tacit knowledge are identified as further important elements of agile software development (Boehm & Turner, 2003).

Many of the principles behind the agile software development methods are not claimed to be new. Several of these ideologies and related agile software development methodologies have roots in, for example, the preceding iterative methodologies (Abrahamsson, 2001) and agile and lean industrial product development (Poppendieck & Poppendieck, 2003). In addition, it has been widely acknowledged

prior to the agile movement that the different methods of software development are far from being neutral and universally applicable (Malouin & Landry, 1983). Benington, among many others, has earlier considered top-down programming and specification as highly misleading and dangerous, as it assumes that sufficient detailed knowledge is available up-front to precisely know the objectives before producing a single line of code, and because it erroneously parallels the software development to the manufacturing industry (Benington, 1983). Furthermore, the positive effect of regular employee involvement in operating decisions and a high degree of responsibility for overall performance in high team spirit, loyalty, and motivation have also already been recognized among production workers (Deming, 1990). Neither has the iterative or incremental mode of software development been invented only by agile proponents, but it has a long history in software development (Larman & Basili, 2003). However, the agile software development approach has accomplished a novel mixture of old and new software development principles that have been gaining increasing interest among practitioners and researchers alike. Williams and Cockburn suggest that the novelty of agile software development is, .if anything, the bundling of the techniques into a theoretical and practical framework. (Williams & Cockburn 2003, p. 40)

In conclusion, the fundamentals of agile software development propose a very different view to the certainty aspect in the software development process, compared to the plan-driven approaches. In agile software development, the uncertainty of schedule, scope and budget of any software development project can be considered as a baseline assumption. Thus, agile software development methodologies can be regarded as a means of responding to the uncertainty of software development, rather than as a means of achieving certainty.

Current Status of Agile Software Development

Currently, there is considerable discussion in scientific forums, both in favour and against agile methodologies. The early agile methodologies, especially, received criticism for the lack of scientific evidence (Abrahamsson, 2002), and their suitability only for software development contexts where small teams were producing non-safety-critical products with volatile requirements (Williams & Cockburn, 2003)

Since the early days of agile software development, an increasing amount of interest has been paid to agile methods, by both practitioners and researchers, thus creating a growing body of empirical data on the different aspects of agile software development. Apart from the individual methods and practices of agile software development, problematic issues have arisen, such as the scalability of agile software

development for large and multisite projects (Eckstein 2004 ; Lindvall et al., 2004) and the compatibility of agile methods with existing standards (Lycett et al., 2003; Paulk, 2001; Reifer, 2003). Recently, the organizational and business aspects of agility have been receiving more attention (Baskerville et al., 2005; Coplien & Harrison, 2005; Oleson, 1998). Accordingly, the early agile methods and techniques have been evolving and are being updated. e.g., XP (Beck & Andres 2004), Scrum (Schwaber, 2004; Schwaber & Beedle, 2002), Crystal (Cockburn, 2005), Test-Driven Development (TDD) (Beck, 2003), and DSDM (Staplenton, 2003)

At present, the most empirical evidence on the agile methodologies problem in adopting agile methodologies can be found in balancing the currently dominating engineering ideologies and methodologies of manageable, predictable and repeatable processes with agile software development methods, which again embrace self-organization, process adaptation and constant changes (Lycett et al., 2003). Balancing the two approaches has been suggested in order to benefit from their strengths, and to compensate for their weaknesses (Boehm & Turner, 2003).

Also there has been some confusion regarding the relationship between unplanned coding and agile software development. It has been proposed that one reason for this confusion is the piecemeal approach of agile software development (Highsmith & Cockburn, 2001). For instance, quality in design in agile software development is prioritized in ongoing design done in smaller chunks instead of massive up-front design of the system (Highsmith & Cockburn, 2001). In fact, the existing agile methodologies, such as Scrum for agile project management and XP for implementation of software, all seem to propose a rather disciplined approach to conducting the tasks of software development (Kähkönen & Abrahamsson, 2000; Nawrocki et al., 2001; Paulk, 2001) Studies indicate that by adopting different agile methods and practices, individual agile software development teams can accomplish a methodology that meets with the goals of CMMI level 2. However, there still seems to be a need to extend agile methodologies in order to meet, for example, CMMI requirements related to more organizational level practices.

Some Common Agile Techniques

Extreme Programming (XP)

The most widely used approach to agile SW development is extreme programming. Communication, simplicity, feedback, courage and respect are the five values that provide foundation to XP. (Pressman, 2010) To achieve simplicity XP restrict developers to design only for immediate needs, rather than consider future needs with the intent to create a simple design that can be easily implemented in code. If the design has to be improved it can be recaptured at a later time. (Pressman, 2010)

XP uses an object oriented approach as its preferred development paradigm and encompasses set of rules and practices within its four frame work activities: planning, design, coding and testing. (Pressman, 2010)

(Source: Pressman 2010, p80)

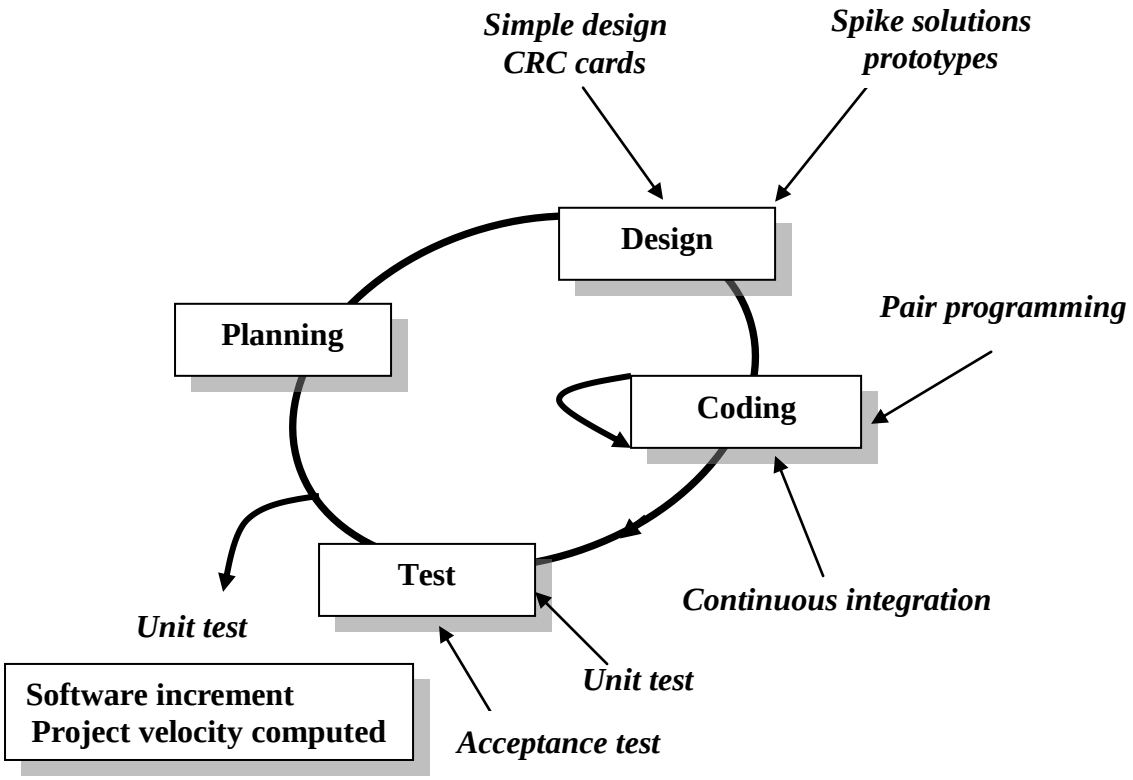


Figure 4: Extreme Programming

Adaptive Software Development (ASD)

ASD has been proposed by Jim Highsmith as a technique for building complex software and systems. ASD mainly focus on human collaboration and team self-organization. ASD life cycle incorporates three phases: speculation, collaboration and learning. (Pressman 2010)

(Source: Highsmith, 1997, p.67)

Figure 5: Adaptive SW Development

Scrum

The name Scrum is derived from an activity that occurs during a rugby match. This method was conceived by Jeff Shutherland and his development team in the early 1990s. Recently further development on the Scrum method has been performed by Schwaber and Beedle. (Pressman, 2010)

Scrum emphasis use of a set of software process patterns that have proven effective for projects with tight time lines, changing requirements and business criticality. Scrum consists of set of development actions as described below:

Backlog: A prioritized list of project requirements or features that provide business value for the customer. Items can be added to the backlog at any time. This is how it incorporates the changes. The project manager set the priorities of the items in the backlog

Sprints: Consists of work units required to achieve a requirement defined in the backlog this has to fit into pre defined time –box. During a sprint changes are not introduced. Sprint enables the team to work in short but stable environment.

Scrum Meetings: These are short (typically 15 minutes) meetings held daily by the team. Three main questions are asked and answered at these meetings:

What did you do since the last scrum meeting?

What obstacles are you encountering?

What do you plan to accomplish by the next team meeting?

The tame leader named scrum master lead the meeting and assess the responses from each member. These meetings help to uncover potential problems as early as possible.

Demos: Use to deliver the SW increment to the customer so that the functionality that has been implemented can be demonstrated and evaluated by the customer. The demo

may not contain all planned functionality, but those functions that can be delivered within the time-box that was established. (Pressman, 2010)

(Source – Schwaber & Beedle 2004, p.54)

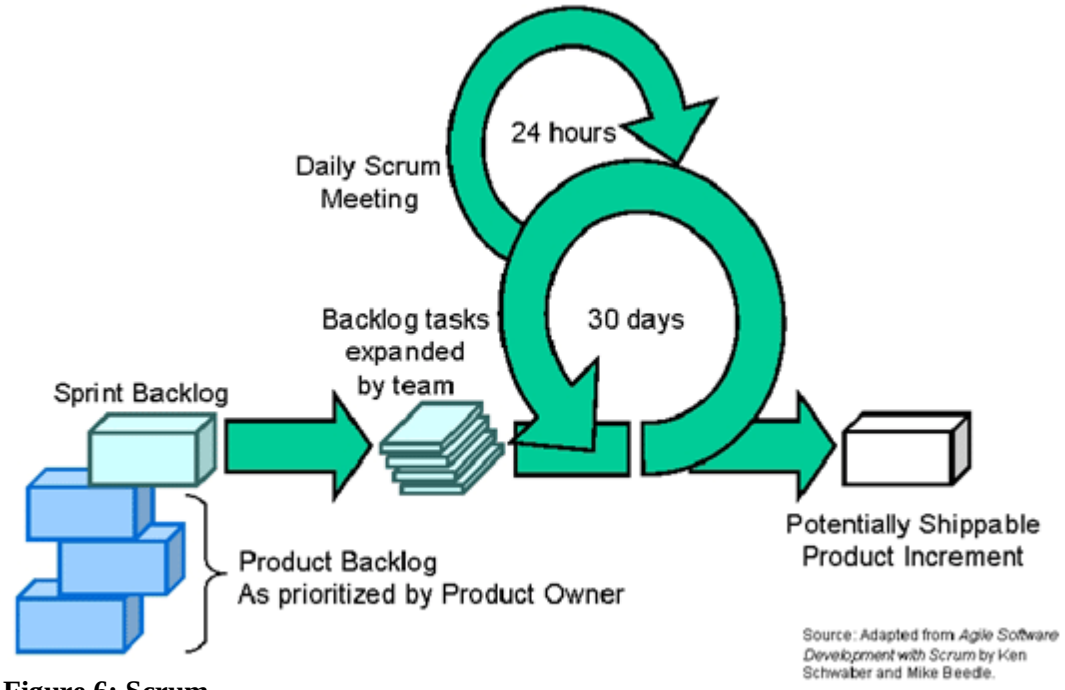


Figure 6: Scrum

2.7 RELATED STUDIES

Because agile development is such a new topic it was very difficult to find related researches. Some analysis was found, but most of the articles and books about the agile development are written by the inventors of agile manifesto and since there can be some sort of biasness toward their own methods the researcher of this thesis has not made much attention on those.

Over view of Agile Management and Development Methods – Addicam.V.Sanlay

The white paper helped the researcher to identify the current software management and development environment, circumstances of the current situation, identify current management and development methods and ascertain how current management and

development processes can handle the problem. Finally it has provided a comparison of agile and traditional methods for a selected software product.

According to author of the above white paper, today, there is a constant need for delivering more in a given amount of time (Sanjay, 2005). In addition, other attributes of the current development environment identified by the author are listed below.

- Availability of skilled professionals - the newer the technology, tools, methods, and domain, the smaller the pool of skilled professionals.
- Stability of implementation technology - the newer the technology, the lower the stability and the greater the need to balance the technology with other technologies and manual procedures
- Stability and power of tools - the newer and more powerful the development tool, the smaller the pool of skilled professionals and the more unstable the tool functionality
- Effectiveness of methods - what modeling, testing, version control, and design methods are going to be used, and how effective, efficient, and proven are they
- Domain expertise - are skilled professionals available in the various domains, including business and technology (Sanjay, 2005). Due to this situation today many of the development processes are uncontrolled, the inputs and outputs are either unknown or loosely defined, the transformation process lacks necessary precision and the quality control is not defined. (Sanjay, 2005) Therefore, today it is difficult to emphasis on process and upfront planning hence “heavy” or “monumental” models like waterfall and spiral which focus primarily on planning is not suitable to cater to the today’s ever changing environment. (Sanjay, 2005)

“Currently, most software management & development is considered a “chaotic” activity, better known as “code and fix”. This mean software is written without much

of an underlying plan, and the design of the software system is cobbled together." (Sanjay, 2005) As Sanjay has described in his white paper the unpopularity and unsuitability of these traditional "heavy" methodologies have lead companies to move for modern methods like agile.

In its second section the white paper describes the history, characteristics of agile development, where to and where not to use agile, currently available agile development processes like extreme programming, scrum, crystal, dynamic systems development method (DSDM) and wisdom. This helped the researcher to sharpen knowledge in agile development.

History of Agile Development

In February of 2001, a group of people, frustrated with the existing heavy software methodologies met in Utah to find some common ground in alternate software development. They came up with this manifesto:

We are uncovering better ways of developing software by doing it and helping others does it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more. (Highsmith, 2001) This manifesto is the cornerstone of all the different Agile Software Management & Development methods.

Agile Development Characteristics

Today's time-sensitive business climate requires that we quickly accommodate requirements changes during development and, after development, be equally adept at delivering the upgrades caused by software's rapid software evolution and the customer's ever-increasing requirements (Aoyama, 2000). A dominant idea in agile development is that the team can be more effective in responding to change if it can:

- Reduce the cost of moving information between people
- Reduce the elapsed time between making a decision to seeing the consequence of that decision
- Place people physically closer
- Replace documents with talking in person and at whiteboards, and
- Improver the team's amicability-its sense of community and morale- so that people are more inclined to relay valuable information quickly
- Make user experts available to the team or, even better, part of the team and
- Work incrementally (Cockburn et al., 2001)

Where to use Agile Development

Agile Management & development methods are used under the following circumstances:

- Customers/users are active participants in requirements and/or analysis modeling efforts
- Changing requirements are welcomed and acted upon accordingly – there is no "requirements freeze"

- Working on the highest priority requirements; first, as prioritized by your project stakeholders, and in turn focusing on highest risk issues as work progresses
- Taking an iterative and incremental approach to modeling
- Primary focus is on the development of software, not documentation or the models themselves
- Modeling as a team where everyone's input is welcome
- Actively trying to keep things as simple as possible – using the simplest tools available and creating the simplest model(s) that do the job
- Discarding most, if not all, models as development progresses
- Customers/business owners make business decisions, developers make technical decisions.
- The content of your models is recognized as being significantly more important than the format/representation of that content
- Test what you are describing with your model(s) is a critical issue being continually considered as you model (Ambler, 2001)

Where not to use Agile Development

Agile development methods are not used under the following circumstances:

- Goal is to produce documentation, such as a requirements document, for sign-off by one or more project stakeholders

- Using a case tool to specify the architecture and/or design of software BUT not using that specification to generate part or all software

- Customers/users have limited involvement with your efforts. For example they are involved with initial development of requirements, perhaps available on a limited basis to answer questions, and at a later date will be involved in one or more acceptance reviews of your work

- Focusing on a single model at a time. Common examples are "use case modeling sessions", "class modeling sessions", or "data modeling sessions." The root cause of this problem is typically "one artifact developers" such as people specialized in data modeling or user interface modeling – with Agile Method generalists should be leading the effort.

- Working towards a freeze of one or more models – In other words you are taking a serial approach.

- Delivering models and/or documentation to another team who will then evolve the system further. In other words, "handing off" work in a serial manner

The third section describes few companies that have moved to agile methods and a comparison of the projects before and after adapting agile techniques. This helps the researcher to get an idea how certain project characteristics have been by moving towards agility. But this has not discussed the difference of cost, time and number of defect factors before and after adaption.

Table 01: Agile Adoption & percentage of Change (Source: Sanjay, 2005)

	Before adapting Agile	**After adopting Agile**	**% Change**
Total Code size	45773	15048	-67%
Average methods per class	6.30	10.95	+73%
Average lines per method	11.36	5.86	-48%
Average cyclometric complexity	3.44	1.56	-54%

This white paper helps the research to sharpen the knowledge in the subject area and provides an answer to the first research question "Why today's companies are rapidly moving into agile software development?" But it has not focused on the main research question, "Whether the companies were able to achieve required process, project and product quality through agility?"

Agile Systems Development and Stakeholder Satisfaction:

A South African Empirical Study - Carlos Ferreira & Jason Cohen

The above named research was implemented using 59 South African Development projects. The aim of the research is to identify how five important characteristics of agile method (namely, iterative development, continuous integration, collective ownership, test-driven design and feedback) influence stakeholder satisfaction. (Ferreira & Cohen, 2008) To facilitate research Ferreira and Cohen has built a conceptual model (figure: 17) and hypothesis. Within the study the stakeholder satisfaction has been considered in terms of satisfaction with process and satisfaction with outcome. The independent variables of the research are the above mentioned five dimensions of agile development. Ferreira and Cohen have briefly explained each of the five characteristics as illustrated in the below Table 2. These details have helped the researcher to enhance knowledge in the subject area.

(Source: Ferreira & Cohen, 2008)

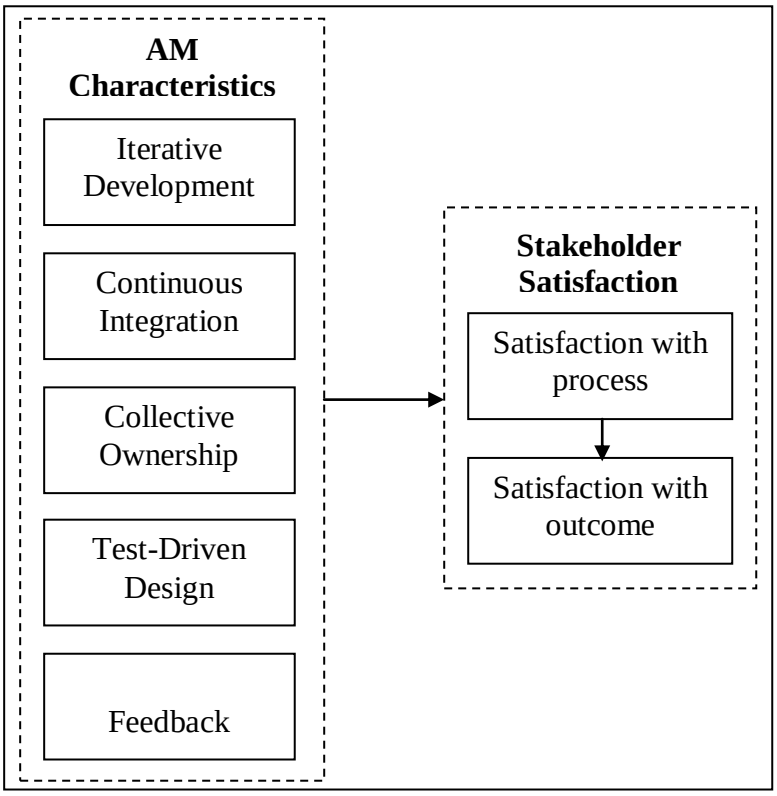


Figure 7: AM Characteristics

Table 02: AM Characteristics (Source: Ferreira & Cohen, 2008)

AM Characteristics	Description
Iterative Development	Quick delivery of small working software releases at regular intervals or cycles
Continuous Integration	New code is integrated in to the production base code continuously, ideally after each task is completed.
Collective Ownership	Any developer has the right to add or maintain the code anywhere in the system at any time
Test-Driven Design	Developers write tests before they code. This practice aims to encourage developers to think before coding
Feedback	Frequent feedback loop with customers allows developers to ascertain the accuracy of the functionality.

Ferreira and Cohen have also described the differences between traditional models and agile development. This helped the researcher to clarify understanding on both methods. Some of the differences noted by Ferreira & Cohen are as follows;

Table 03: Traditional vs. Agile Development (Source: Ferreira & Cohen, 2008)

Traditional Methods	Agile Development
Process oriented, life cycle base and plan driven with heavy documentation	No upfront planning and heavy documentation
Focused on optimized processes	Short iterative cycles of development based on product features.
Neither rapid feedback nor changes for the system under development	Collaborative decision making, incorporation of rapid feedback and change, and continuous integration of code changes into the system under development
Plan projects around tasks and documentation	Plan projects around features where development is evolutionary and iterative
Fix functionality and then adjusts time and resources to reach the functionality	Fix time and resources, and then adjust the amount of functionality

The research done by Ferreira and Cohen has focused on agility and customer satisfaction where as the project focuses on identifying the quality gap between traditional methods and agile development. Since customer satisfaction and software quality related to each other this has provided direction to the researcher to carry out the research on quality and agility, even though the researches have different aims and methodologies. Ferreira and Cohen have done their study in the context of South Africa, where as this research focuses on the Sri Lankan software development industry. Ferreira and Cohen have carried out a hypothesis testing and developed five hypotheses to achieve research objectives. Though the above mentioned study is to find the customer satisfaction on some important agile characteristics, the research aimed at Investigating whether there is a quality difference between agile and traditional methodologies and in accentuating the difference.

Agile Modeling (AM) – Using Models to Carry Out the Development Process

By Scott Abler, 2002

"Using agile modeling techniques and tools allows software developers to consider complex problems before addressing them in programming. Agile planning and development uses software modeling principles to let a developer design a software system that truly meets the customer's requirements. This will lead to develop a final product capable of catering the user's expectation" (Ambler, 2002). As described above according to Scott Ambler who offered a suite of principles and practices for software modeling, it is easy to meet user expectations or customer requirements through Agility. According to Scott following factors make a contribution to the high achievement of user expectations in Agile development.

- Stakeholders actively participate in the agile planning and development
- Teamwork is established
- Appropriate artifact (such as UML diagrams) is used to create suitable models
- Several models are created in parallel
- Correctness of the agile software models is verified
- The verified models are implemented and the resulting interface is presented to the user
- Standards for agile requirement management are met

Also the values of the *agile software development methodology* have pointed out by Scott as stated below;

- Communication – the agile team can exist and do tasks only in case communication channels are established. The project manager is a person who ultimately cares for establishing and maintaining the channels. Agile modeling entails using agile project management software to build such channels and let the team communicate with each other in real time.
- Simplicity. Agile planning and development will let the project team to make their effort simple yet complete. Simple project management can be achieved if developers clearly know their roles and duties within the agile modeling project, and there are no nodes that make team collaboration and data exchange more complicated.
- Rapid Feedback – the agile development process becomes effective if team members request and give feedback. The agile project manager can receive feedback to analyze issues and implement solutions that best contribute to the achievement of balanced agile estimating and planning.
- Humility means a developer understands that he or she may not know everything about the project so he/she should collaborate with the team to share knowledge and perceive software development ideas. Following this principle is a great contribution to the agile development management.

But in his study he has not considered traditional methods nor had compared Agile with Waterfall or any other traditional method. But the aim of this research is to indentify whether there is a difference in software quality between the products developed using Agile and traditional methods. (Ambler, 2002)

An Introduction to Agile Software Development

By Szalvay, 2004

The article begins with describing the two development methods and further it moves toward to contrast the two methods. As been described in the article one of the most important differences between the agile and waterfall approaches is that waterfall features distinct phases with checkpoints and deliverables at each phase, while agile methods have iterations rather than phases. The output of each iteration is working code that can be used to evaluate and respond to changing and evolving user requirements.

Waterfall assumes that it is possible to have perfect understanding of the requirements from the start. But in software development, stakeholders often don't know what they want and can't articulate their requirements. With waterfall, development rarely delivers what the customer wants even if it is what the customer asked for.

Agile methodologies embrace iterations. Small teams work together with stakeholders to define quick prototypes, proof of concepts, or other visual means to describe the problem to be solved. The team defines the requirements or the iteration, develops the code, and defines and runs integrated test scripts, and the users verify the results. Verification occurs much earlier in the development process than it would with waterfall, allowing stakeholders to fine-tune requirements while they're still relatively easy to change. (Szalvay, 2004)

As stated above the article discuses the practical differences in the in methodology used in Agile and Waterfall methods. But in contrast this study aims at identifying the software quality difference between the products developed using these methods.

Agile Adaption Rate Survey result: February 2008

By Jon Erickson, 2008

This survey was performed in early February 2008 using 642 respondents. The survey was announced by Jon Erickson, the editor of the Dr. Dobb's Journal.

The findings of the survey stated:

1. 69% of respondents indicated that their organizations are doing one or more agile projects. Of those that hadn't yet started, 15% believed their organizations would do so within the next year.
2. 61% of developers think that their orgs are doing agile, whereas 78% of management thinks so. Apparently developers are a bit more discerning.
3. 82% of organizations doing agile were beyond the pilot project phase.
4. Respondents overwhelmingly indicated that agile teams are producing higher quality, have greater productivity, and enjoy greater stakeholder satisfaction. See Figure 2.18.
5. Agile success rates: 82% for co-located teams, 72% for near located (people in different cubes, on different floors, working from home), 60% for significantly distributed (planes would be involved to get people together). See Figure 2.19
6. 84% of agile teams have iteration lengths of 4 weeks or less, and 2 week iterations are the most popular.
7. Although on average the costs are lower on agile teams, 23% of respondents believe they are experiencing higher average costs. 40% said costs were unchanged and 37% had lower costs.
8. Co-located agile projects are more successful on average than non-co-located, which in turn are more successful than projects involving off shoring.

(Source: Ambler, 2007)

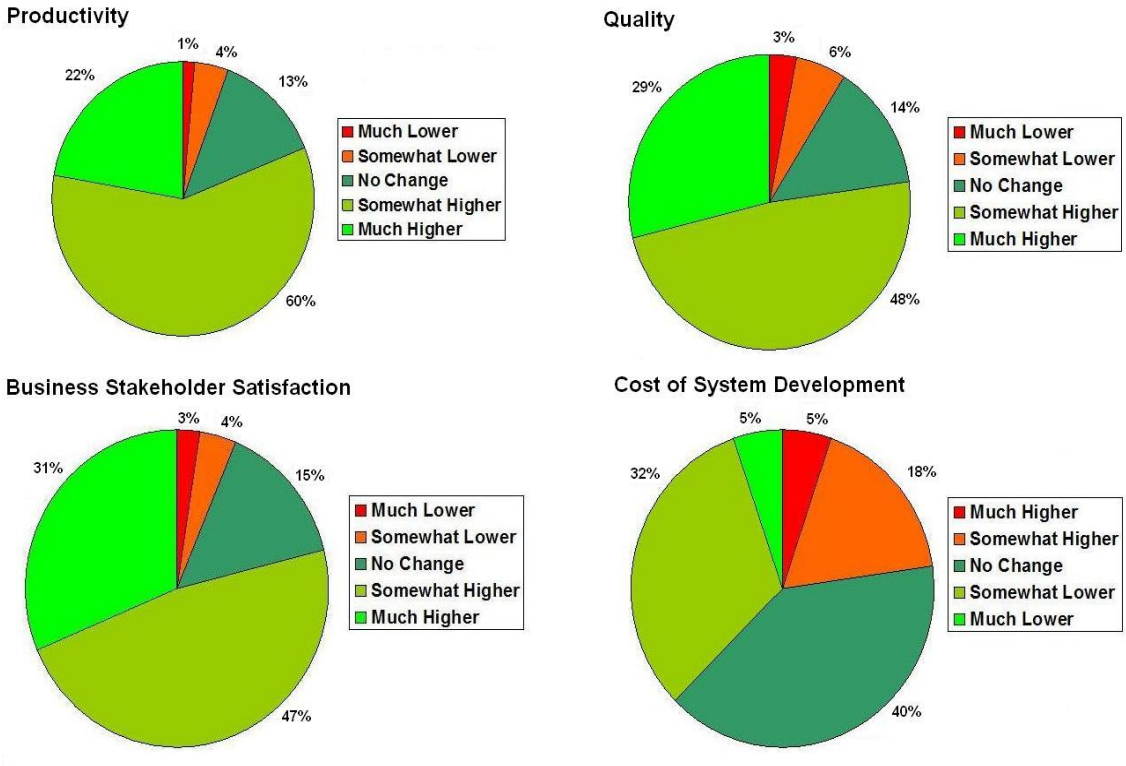


Figure 8: Effectiveness of agile software development compared with traditional approaches.

(Source: Ambler, 2007)

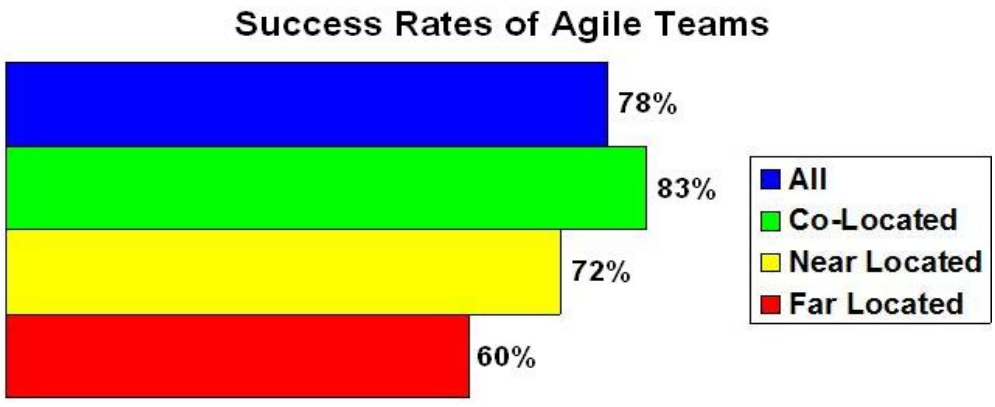


Figure 9: Agile success rates by level of team member distribution

Waterfall Model Vs Agile

By Gray Pilgrim, 2010

This article is a comparative analysis of waterfall model vs agile model of software development. According to the article two of the most popular software development models are the 'Waterfall Model' and the 'Agile Model'. This article makes a waterfall model vs agile model comparison that will serve to point out the differences in the two different methods of software development. This comparative analysis may help in choosing which model is conducive for software development project.

The article discusses the differences between Waterfall Vs Agile in terms of Efficiency, Suitability and Conceptual differences. (Pilgrim, 2010)

Efficiency: The author of this article has determined the efficiency by the quality of ultimate software product, number of bugs and the development time consumed. The finding of the Pligrim's study has reflected that the Agile is more efficient than Waterfall due to its adaptability to the real world. (Pilgrim, 2010)

Suitability: According to the findings of the above study Waterfall model is suited for development of programs that are already stable where the design does not need a major alteration. Agile development is more suitable at situations where the requirements are changing rapidly. (Pilgrim, 2010)

Conceptual Differences: Waterfall model as its name reflected has a sequential process in software development. Just like water progressively falls from one altitude to the lower in a waterfall the production cycle of a software progress sequentially in the Waterfall model. Where as in agile breed of models focused on 'agility' and 'adaptability' in there development.

Improving Software Quality with Agile testing

By Sugnadhi AD et al., 2008

The article published in the International Journal of Computer Applications stated "Software engineering can be complex and hence has the risk of project delays, defective product due to time constraints that lead to the risk of losing projects from the customer. To succeed it demands the use of IT methodologies. But off the shelf methodologies are not flexible. QA is an important part of software and ensures the quality of deliverable. Agile development completely redefines quality assurance work—from formal roles to day-to-day activities—making some traditional QA responsibilities and outputs irrelevant. In this paper, we describe application of Agile testing to software quality management that will ease development and testing. Also we will explain how it helps in better planning and time management and how it enables clients to achieve improved coordination of their test resources with the agile development team by allowing automated tests to be developed in tandem with code development on the same set of requirements. We'll summarize with existing agile testing applications, most notably by describing how to effectively use professional testers and how to thoroughly acceptance-test a system that's too large and complex for a single customer to specify and the future of agile testing."

Waterfall Vs Agile: Can they be Friends

By Alberto Gutierrez

In the above mentioned article Gutierrez distinguished between projects suitable for Agile Development and projects suitable for Waterfall development.

According to the article Agile is best suited to projects that:

- **Focus on time to market** - Time to market measures how fast a company can have a product out in the market from the moment they start developing. A fast time to market allows the company to have its product available long before its competitors. Agile is a sure bet to achieve very fast times to market as at the end of each iteration the application should be production ready.
- **May require a high degree of change** - Requirements can and do change for projects as they progress. Agile provides a flexible process that optimizes feedback so changes can be introduced reliably during development.
- **Have one unique important deliverable: the product** - There are many projects that only have one important output, the final product. Agile focuses on the product, almost ignoring other artifacts, such as documentation.

And the Waterfall best suited projects include

- **Are contract based -** The customer requires the company providing the software to commit in writing to fulfill a series of requirements. Since Waterfall is document driven, it lends itself to contracts that are heavily based on requirements. This helps to guarantee that everything specified on the contract is complete.
- **Are focused on analysis -** Some software development projects require the analysis to be completed beforehand; this would be the case of very complex or critical systems that require many validation steps or approvals. Being a sequential process Waterfall is naturally suited to this purpose.
- **Have more than one deliverable -** Not just the product, but also the user manual, the architecture …etc. Waterfall produces documents and artifacts other than the software itself. For some projects, these artifacts are considered almost as important as the final product.

The Agile Impact Report: Proven Performance Metrics from the Agile Enterprise

By Toronto & Boulder, 2008

The article discusses the advantages of Agile development in terms of 'Time to Market', 'Productivity' and 'Number of defects'

Under the subheading Time to Market it has stated the following;

Larger software development teams, especially when geographically dispersed, often struggle to deliver their software on time. By adopting Agile practices, companies measured in this study were able to produce large-scale enterprise software in four to eleven months, compared to the six to thirteen months a typical organization required to deliver comparable software. Overall, Agile companies experience an average increase in speed of 37 percent. The customers who participated in the study saw an average increase of 50 percent in their time-to-market when compared to the industry average.

2.7 CONCLUSION

According to the literature presented in this chapter Agile vs Waterfall plays a major role in today's software development industry. It seems that no IT meeting/discussion ends up without comparing Agile and traditional methodologies. In order to keep pace in the global competitive market most of the IT companies are looking forward to adopt cost and time effective development methods. As stated in the literature unlike the traditional methods Agile techniques facilitate number of incremental releases. Hence software development companies are rapidly moving in to Agile development due to the factor Time to market. But the argument on the other hand was though this method facilitate to release working products in short cycles can they achieve the

expected level of quality through Agile development in terms of various different quality factors introduced by different quality models and quality management experts.

When considering the related researches the researcher of this thesis observed that there are vast number of researches done on agile software development and its advantages. But none of them conducted in the Sri Lankan context. In addition, there were very few researches done to check quality difference between Agile and Traditional methods except the factors time to market and cost effectiveness.

CHAPTER THREE

3. METHODOLOGY

3.1 INTRODUCTION

This chapter provides the road map (outline) to achieve the research objectives. The chapter presents the research design by describing the research approach, target population, sample size, sampling technique, data collection and data analysis methods.

3.2 RESEARCH APPROACH

It is the observation by the researcher of this paper that majority of the software development companies in the industry today are rapidly moving towards Agile development. As explained in the literature, the main reason for this diversification is the ‘time to market’ and ‘feasibility to incorporate requirement changes even in the later stages of development’ (Williams & Cockburn, 2003) According to the industry professionals this is very important for their survival in the modern rapidly changing environment. But the puzzle with the researcher of this thesis is whether the companies are able to achieve expected software quality through Agile development and to ascertain the most appropriate agile technique in the Sri-Lankan context.

In the process of seeking answers to the above question the researcher initiated the following steps:

Step1: Feasibility study to check the possibility of collecting relevant information.

The researcher identified the major software development companies in the island and considered the IT companies providing software development services and registered with the Software Exports Association. There were 27 such companies as of June 2009 (Figure 01). Then the researcher sent e-mails to these companies inquiring their ability to provide information for the above research. Out of them only seven companies agreed to provide data.

Then the researcher designed a preliminary questionnaire to investigate the feasibility of obtaining relevant information for the research (See Appendix A). Only four companies were selected for this research based on the responses provided in the feasibility study. The selected companies are Virtusa Pvt Ltd, Teamwork Technologies, DMS Software and E-College.

The researcher then conducted another study to find out the different agile techniques in usage in the selected companies (See Appendix B). Unfortunately the result reflected that only one company is using Scrum, ASD and XP techniques. All the other companies were using only Scrum. Hence, the researcher found that it is impossible to identify the most suitable agile technique for Sri Lankan context at present.

Based on the feedback for the pilot surveys conducted, the researcher decided to confine this research to find the answer to the research problem: *"Is there a quality difference between the Software products developed using Agile and Traditional methods? And if there is a difference what is the healthier method? "*

Step 2: discovery of software quality factors and identification of quality factors to be considered at the research.

From the literature survey the researcher has identified set of quality factors via different philosophies of quality after considering the view points of quality management gurus and various quality models.

Through a preliminary investigation the researcher identified that it is not possible to obtain client information; therefore it was impossible to measure software quality from the user's perspective. (See Appendix A Q7) Due to such difficulties the researcher confined this research only to developer oriented (Organizations view point of) quality attributes.

Therefore, it was decided to take the blending of organization related quality attributes from all three popular models referred to in the previous chapter.

It is not an easy task to differentiate developer oriented quality attributes from user oriented attributes, as quality classifications are different from each model, and some attributes are subjective to multiple definitions.

According to the analysis done by Berender on 'Software Quality Attributes and tradeoffs', the following criterion in chapter 4 has selected as developer oriented attributes to be consider in the current study:

Correctness

Testability

Changeability / Stability

Install ability (Berander et al., 2005)

As per the literature survey, quality software is a software product which is reasonably defect-free, delivered on time and within budget, meets requirements

and/or expectations (Raman, 2009). Therefore, the number of defects, time and cost factors are also considered in the research. As described in the previous chapter time and the cost factors are considered under project quality, where as the number of defects are considered under product quality. According to Raman all the above identified quality factors on the other hand affects the defects ratio.

The following section describes in detail the quality factors selected. For consistence interpretation of the quality attributes, the definition of attributes have been used according to Software Engineering Institute's (SEI) Software Technology Road Map Glossary (SEI:Glossary, 2011) and ISO 9126 definitions (ISO/IEC 9126, 1991)

Correctness

"The degree to which a system or a component free from faults in its specification, design, and implementation" (SEI:Glossary, 2011)

Testability

"The degree to which a system or a component facilitate the establishment of test criteria and the performance of tests to determine whether those criteria have been met" (SEI:Glossary, 2011).

Changeability

"The capability of the software product to enable a specified modification to be implemented" (ISO/IEC 9126, 1991)

Stability

"The capability of the software product to avoid unexpected effects from modifications of the software" (ISO/IEC 9126, 1991)

Install ability

"The capability of the software product to be installed in a specified environment" (ISO/IEC 9126, 1991)

Having identified the variables, the following relationship have been derived based on the definitions and reviewed literature:

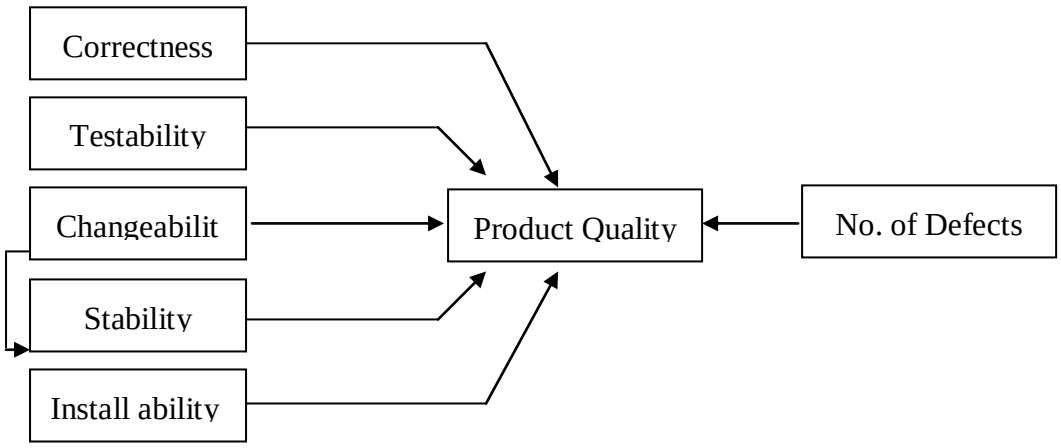


Figure 10: SW Product Quality

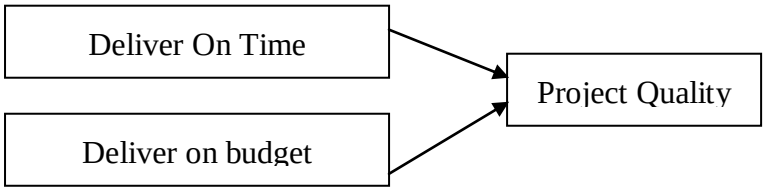


Figure 11: SW Project Quality

In the Berander's analysis he has acknowledged set of quality attributes for each of selected quality factor. Quality factor and corresponding attributes are listed below;

- Correctness – Compliance Specifications, Uniformity of Functionality, Defects per Kilo of Lines of Code (KLOC)
- Testability – Test Coverage Percentage, Test Effectiveness Ratio, Simplicity, Coding Standards.
- Changeability / Stability – Ability to modify, Errors after modifications, Effoer for modifications, Level of Cohesion and Coupling
- Install ability – Stability level of the system after installation

Based on the identified quality factors and the attributes of each factor following conceptual frame work was derived;

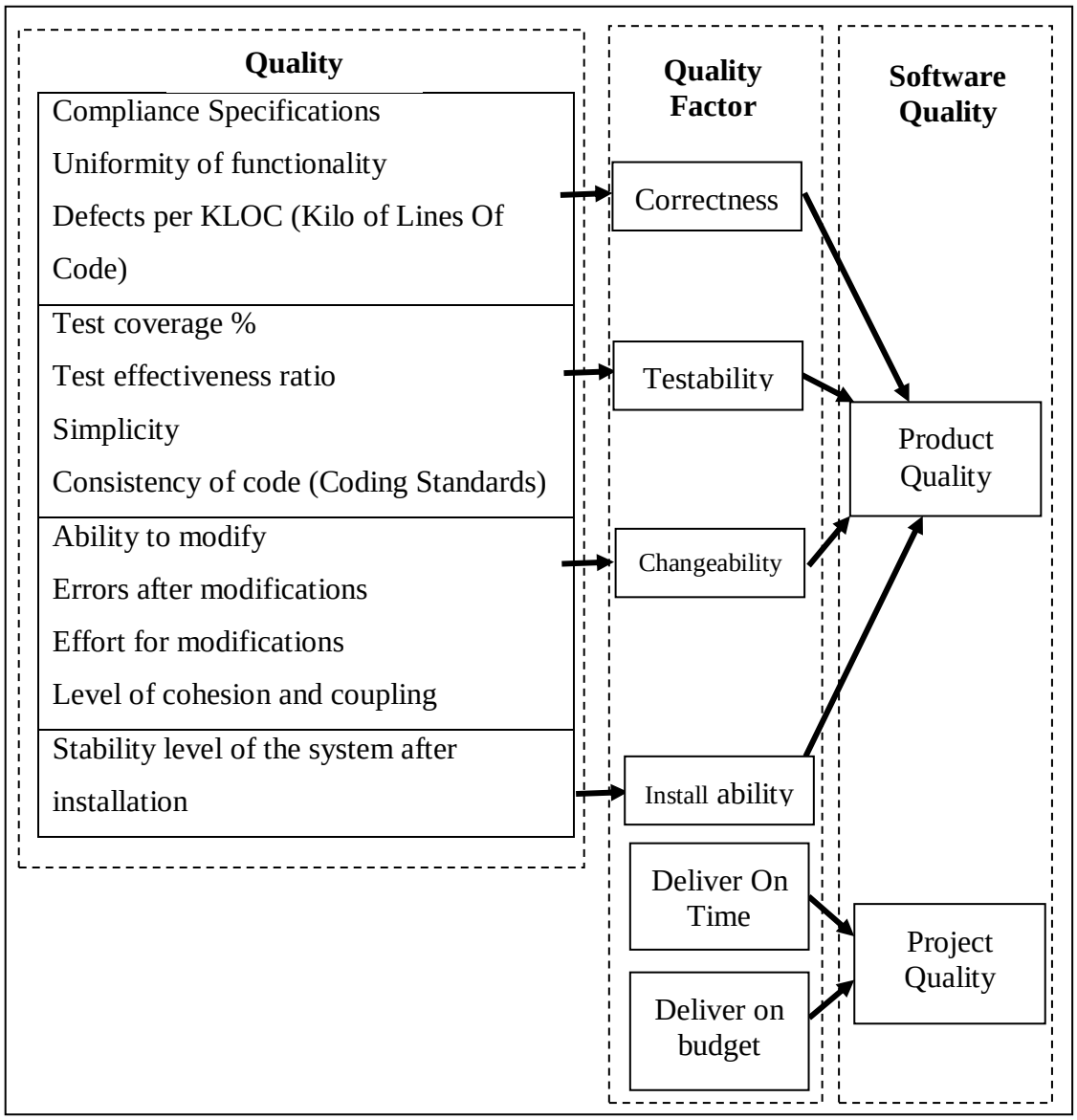


Figure 12: Conceptual Frame Work

3.3 POPULATION

Since the study is based on software development organizations, the research has focused on companies registered with the "Software Exports Association Sri Lanka". There were 31 registered members with the association as of 1st of July 2009 and out of it 27 companies are engaged in software development. (Figure 01) According to the feasibility study (Described in page 72, step1) conducted, only 4 companies out of 27 were feasible for the research. These four companies were selected based on the

capability to collect relevant information. (See Appendix A) The target population for the research includes Project managers, Developers, Quality Assurance team leads and Testers. Employees under the above mentioned designations were selected because they can directly provide the required information for the research. The table given below (Table 4) summarizes the number of employees in each company in selected professions (Appendix C)

Table 04: Population

Company Name	Number of personals as				Total
	PM's	Developers	QA Leads	Testers	
Virtusa Corporation	62	650	75	250	1040
Team Work Technologies	06	40	02	05	53
DMS Software Technologies	01	30	01	05	37
E- Colledge	05	70	06	25	106
Total	74	790	84	285	1233

3.4 SAMPLE SIZE

Morgan's theory (Krejcie & Morgan, 1970) is used to decide the sample size separately for each company. The research applies the theory to the total number in the population (company wise) and decides the total sample size for each company. Then it has calculated the amounts for each category proportionately (Table 5). If the value is with decimal points it is rounded out to the next highest number. The research has taken a confidence interval of 95% to 99% and the error term is 0.1 when applying the theory.

Note: If there is only one employee under a given category the employee is taken for the sample without applying the theory.

Table 05: Sample

Company Name	Number of personals as				Total
	PM's	Developers	QA Leads	Testers	
Virtusa Corporation	05	54	06	22	88
Team Work Technologies	02	26	02	04	34
DMS Software Technologies	01	22	01	04	28
E- Colledge	03	34	03	12	52
Total	12	136	12	42	203

3.5 SAMPLING TECHNIQUE

Random Sampling technique is used to select the sample with the given inclusive criteria

- Respondent must have at least one project completed in each method.

The researcher handed over the questionnaires or conducted interviews only with individuals who satisfied the above criteria.

3.6 DATA COLLECTION

The research collected data through interviews and questionnaires prior to submitting the questionnaire or conducting interviews, the researcher made sure that the randomly selected individual satisfied the above mentioned inclusive criteria.

The researcher conducted interviews with the project managers to collect in depth and in detailed information. The interviews mainly focus on capturing the project specific demographics. These demographics include time and the cost factors. The semi structured interviews facilitated the research to gather answers not only for set of pre defined questions, but also for some important hidden information.

As already mentioned above, the research collected data not only through interviews, but also by providing questionnaires to QA leads, Developers and Testers. The questionnaires were distributed mainly via e-mails and helped the research to obtain information from a large number of personnel. The questionnaire focused on capturing details on the following areas:

1. Correctness
2. Testability
3. Changeability
4. Install ability

Below table summarizes the questionnaire design (See Appendix D)

Table 06: Questionnaire Design

Quality Factor	**Measure \ Attribute** (Berander et al., 2005)	**Questions**
Correctness H01	Compliance Specifications Uniformity of functionality Defects per KLOC (Kilo of Lines Of Code)	Q-05 to Q-09
Testability H02	Test coverage % Test effectiveness ratio Simplicity Consistency of code (Coding Standards)	Q-10 to Q-12 & 15
Changeability/ Stability H03	Ability to modify Errors after modifications Effort for modifications Level of cohesion and coupling	Q-13 to Q-17
Install ability H04	Stability level of the system after installation	Q-18 to Q-20

3.7 DATA ANALYSIS

The aim of this study is to analyse the software quality difference between Agile and Waterfall techniques. And if there is a difference then to identify the technique by which a company can produce better quality software products.

In order to analyze this variation the research used Hypothesis testing and the following Hypothesis was derived considering the selected quality factors.

Hypothesis 1

H0 – There is no difference in Correctness between the software products developed using Agile and Waterfall methods ($\mu AC - \mu WC = 0$)

H1 – There is a difference in Correctness between the software products developed using Agile and Waterfall methods ($\mu AC - \mu WC \neq 0$)

Hypothesis 2

H0 – There is no difference in Testability between the software products developed using Agile and Waterfall methods

H1 – There is a difference in Testability between the software products developed using Agile and Waterfall methods

Hypothesis 3

H0 – There is no difference in Changeability between the software products developed using Agile and Waterfall methods

H1 – There is a difference in Changeability between the software products developed using Agile and Waterfall methods

Hypothesis 4

H0 – There is no difference in Install ability between the software products developed using Agile and Waterfall methods

H1 – There is a difference in Install ability between the software products developed using Agile and Waterfall methods

Hypothesis 5

H0 – There is no difference in Time to Market between the software products developed using Agile and Waterfall methods

H1 – There is a difference in Time to Market between the software products developed using Agile and Waterfall methods

Hypothesis 6

H0 – There is no difference in estimated and actual budgets between the software products developed using Agile and Waterfall methods

H1 – There is a difference in estimated and actual budgets between the software products developed using Agile and Waterfall methods

Based on findings for the above hypothesis finally to come up with,

Hypothesis 7

H0 – There is no difference in software quality between the software products developed using Agile and Waterfall methods

H1 – There is a difference in software quality between the software products developed using Agile and Waterfall methods.

As reflected by the research topic "Software Quality in Agile Development: A gap Analysis between Agile and Waterfall Software Development Methodologies", primarily it is intended to capture the gap between Agile and Waterfall techniques for each identified quality factor. Therefore the research is designed to capture the gap for each question by subtracting the weight received for Waterfall by the weight received for Agile.

When discussing about the weights, the research used 5 point Likert scale that uses the responses Strongly Disagree, Disagree, Neutral, Agree and Strongly Agree. In this research each has been weighed as given below:

1 = Strongly Disagree
2 = Disagree
3 = Neutral
4 = Agree
5 = Strongly Agree

Therefore, the potential differences for each question can vary between ±4. The table below (Table: 7) describes the potential differences and the feasible weighting to reach each difference:

Table 07: Quality Difference

Difference	Agile	Waterfall	Perception
+ 4	5	1	Huge Difference
+ 3	5	2	Moderate Difference
	4	1	
+ 2	5	3	Little Difference
	4	2	
	3	1	
+ 1	5	4	Very little Difference
	4	3	
	3	2	
	2	1	
+ 0	5	5	No Difference
	4	4	
	3	3	
	2	2	
	1	1	
-1	4	5	Very little Difference
	3	4	
	2	3	
	1	2	
- 2	3	5	Little Difference
	2	4	
	1	3	
-3	2	5	Moderate Difference
	1	4	
-4	1	5	Huge Difference

Method of analysis:

- Once all the questionnaires are collected calculate the Agile – Waterfall gap for each question for every respondent.

  Eg: Gap Q5 = Agile Q5 – Waterfall Q5

- Compute each respondents view for each identified quality factor

  Eg: Take the average of the related gaps

  Correctness = Average (Gap Q5, Gap Q6, Gap Q7, Gap Q8, Gap Q9)

- Finally, in order to test the derived hypothesis on Correctness, Testability, and Changeability and Install ability use One Sample T – test on each of the above identified quality factors.

- In order to check the hypothesis on Time and Budget use the Chi Square Test.

The 8 table below summarizes the Analysis techniques with specific objectives.

Table 08: Objective vs. Analysis

Objective	Analysis	Method	Data Collection
1. To identify SW Quality Factors	-	-	Literature Survey
2. To identify SW development process models	-	-	Literature Survey
3. To identify the quality difference between the development methods for identified quality factors	Hypothesis 1	One Sample T Test	Questionnaire Q05 –Q09
	Hypothesis 2	One Sample T Test	Questionnaire Q10 –Q12 & 15
	Hypothesis 3	One Sample T Test	Questionnaire Q13 –Q17
	Hypothesis 4	One Sample T Test	Questionnaire Q18 –Q20
	Hypothesis 5	Chi Square Test	Interview
	Hypothesis 6	Chi Square Test	Interview
4. To identify the most suitable development method in attain each of the identified quality factor	Hypothesis Test	From the confidence interval	-

CHAPTER FOUR

4. ANALYSIS

4.1 INTRODUCTION

This chapter presents and discusses the findings on the research done on Software Quality in Agile Development. In order to achieve its objective this chapter examine the quality gap between software products developed using Agile and Waterfall techniques. As described in Chapter 3: Methodology, the data was analyzed using One Sample T – test and Chi Square test. The chapter starts by presenting the data received with the demographic profile of the responders. It moves to the analysis of each of the quality factor aimed in the research and concludes with a brief summary of key findings.

4.2 DATA RECEIVED

In order to collect data the researcher tendered 190 questionnaires among Developers, Testers and QA Leads. The table below recapitulates the data received.

Table 09: Data Received

	Developers			QA Leads			Testers			
Company	**Smpl**	**Rcvd**	**Acc**	**Smpl**	**Rcvd**	**Acc**	**Smpl**	**Rcvd**	**Acc**	
Virtusa	54	47	20	6	6	6	22	20	18	
DMS	22	18	14	1	1	1	4	4	4	
TeamWork	26	26	26	2	2	2	4	4	4	
E-Colledge	34	22	16	3	2	2	12	3	0	
Total Sample	136			12			42			**190**
Total Receive		113			11			31		**155**
Total Accepted			76			11			26	**113**

Response Rate	81.59%
Valid % against Sample	59.47%
Valid % against Responses	72.90%

Smpl – Sample Size

Rcvd – Total Received

Acc – Accepted Amount (Questionnaires within the inclusive criteria)

As summarizes in the above Table 9 the researcher received 155 questionnaires filled out of total tendered. Therefore, the response rate is around 82%. As cited in the Chapter 3 Methodology the inclusive criteria for the data collected is that 'respondents should fill the questionnaire against both the development methods Agile and Waterfall'. Consequently 42 responses that were not within the inclusive criteria had eliminated. Thus out of 155 questionnaires received, only 113 have been considered in this analysis.

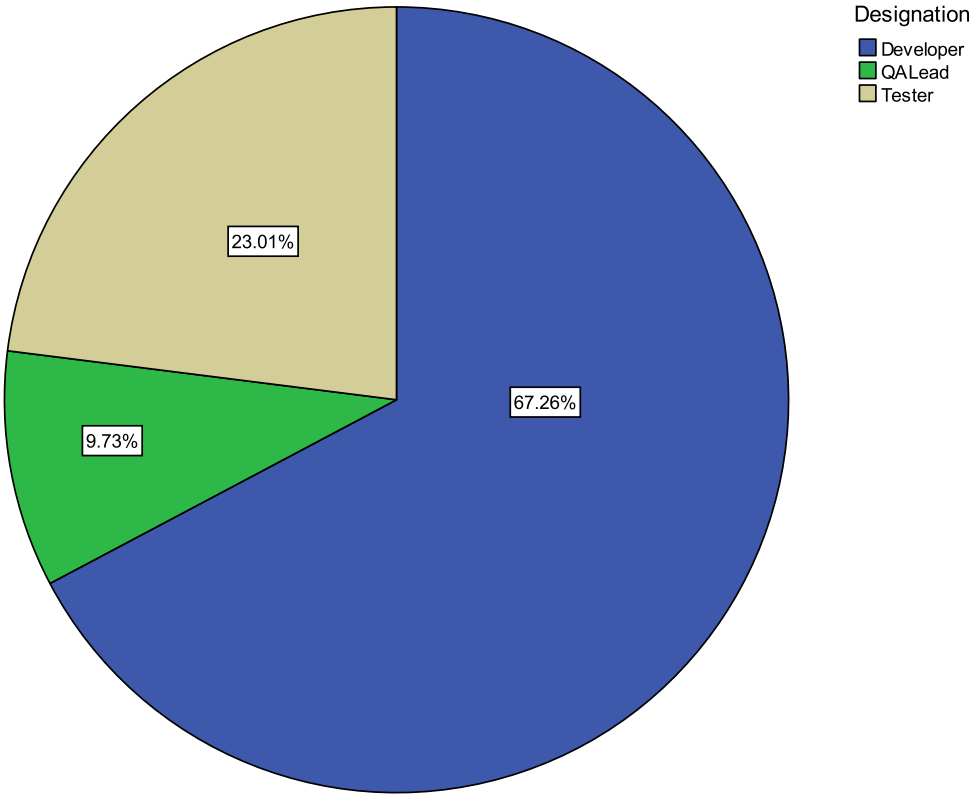


Figure 13: Respondents

The pie chart (Figure 23) above presents the diversification of the 113 responders participating in the study based on their designation. According to the chart out of valid questionnaires 67.3% respondents are developers, 23% are testers and 9.7% are QA leads.

Following rates are calculated by analyzing questionnaires received

Total Values

- Response Rate : 81.58 %
- Valid rate against Sample : 59.47 %
- Valid Rate against responses : 72.90 %
- Total Rejection Rate : 27.09 %

Respondent Categories

- Response rate of Developers : 83 %
- Response rate of testers : 74 %
- Response rate of QA leads : 92 %
- Valid response rate of developers : 67.25 %
- Valid response rate of testers : 83.87 %
- Valid response rate of QA leads : 100 %

4.3 HYPOTHESIS TEST

The second part of the findings presents and analyses the collected data on the identified quality factors such as Correctness, Testability, Changeability, Install ability, Time and Budget. In the process of analyzing each of the above quality factors, this section provides a brief analysis of each question used to describe the identified quality factor.

4.3.1 CORRECTNESS

As presented in Chapter 3: Methodology, questions 5 to 9 in the questionnaire designed to capture the information related to the quality factor – 'Correctness'. This section briefly analyses the responses received for each question and further provide a summary analysis using One sample T – test to verify the derived hypothesis.

Question 5 - The components we deliver almost meet user expectations.

The calculation is done by subtracting the weight receives for Waterfall by the weight receives for Agile. A positive difference implies that Agile is better in meeting user expectations than Waterfall. Where as a negative difference implies that Waterfall is better in meeting user expectations than in Agile development.

Table 10: User Expectation coverage in the final product

Q5. Meet User Expectations 100 %

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	10	8.8	8.8	8.8
	-1.00	10	8.8	8.8	17.7
	.00	49	43.4	43.4	61.1
	1.00	36	31.9	31.9	92.9
	2.00	8	7.1	7.1	100.0
	Total	113	100.0	100.0	

As can be seen in the above Table 10 the calculated differences between Agile and Waterfall for meeting user expectation vary between -2 and +2. Since the variation does not go beyond ±, 2 we can conclude that though there is a variation between Agile and Waterfall in relation to achieving user expectations, no respondents have

perceived a huge difference between the two development methods. (For example no one has rated 1 for Agile and 5 for Water fall or wise versa)

According to the values presented in table 10, out of the total 113 responses received, 17.6 % of responses plunge at negative region and 39 % plunge at positive region. And the majority 43.4 % falls on 0 this means majority of the respondents assume that there is no significant difference between the two development methods for the factor identified by question number 9. As described in chapter on methodology, this can be achieved by selecting the same rating for both development methods (Example 1-1, 2-2, 3-3, 4-4 or 5-5). Out of those who perceive that there is a difference more respondents believe that Agile is better than Waterfall in meeting user expectations.

Question 6 - Our requirement specifications capture all the user requirements.

The calculation is done by subtracting the weight received for Waterfall by the weight received for Agile; a positive difference implies that Agile is better in capturing user requirements than Waterfall and a negative difference implies that Waterfall is better in capturing user requirements than Agile.

Table 11: User expectation Coverage in the Specification

Q6 – Specification capture user expectations

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	14	12.4	12.4	12.4
	-1.00	18	15.9	15.9	28.3
	.00	55	48.7	48.7	77.0
	1.00	12	10.6	10.6	87.6
	2.00	14	12.4	12.4	100.0
	Total	113	100.0	100.0	

According to the values presented in Table: 11 calculated differences between Agile and Waterfall in capturing user expectation varies between -2 and +2. This means though there is a difference between the two methods the gap is small.

As been figured out by the above table, out of 113 total responses the majority of the respondents consider that there is no difference between Agile and Waterfall techniques. There are 55 respondents who support this observation and proportionately it is 48.7 %. On the other hand, 23 % of the responses fall at the positive region and 28.3 % of the responses in the negative region. When considering the negative and positive frequencies there are more respondents toward negative. This indicates that out of those who believe that there is a difference in capturing user expectation, most of them assume that Requirement Specifications in Waterfall method is satisfactory than SRS's in Agile method.

Question 7 - Our system design cover the specifications 100%

As already mentioned, the calculation in this section is done by subtracting the weight receives for Waterfall by the weight receives for Agile. Thus a positive difference implies that in Agile development System design is in 100 % adhere to the specification in contrast to Waterfall. Where as a negative difference implies that Waterfall is better in covering the specification than Agile. Deference equals to 0 means that the respondents have provided the same rating for both methods.

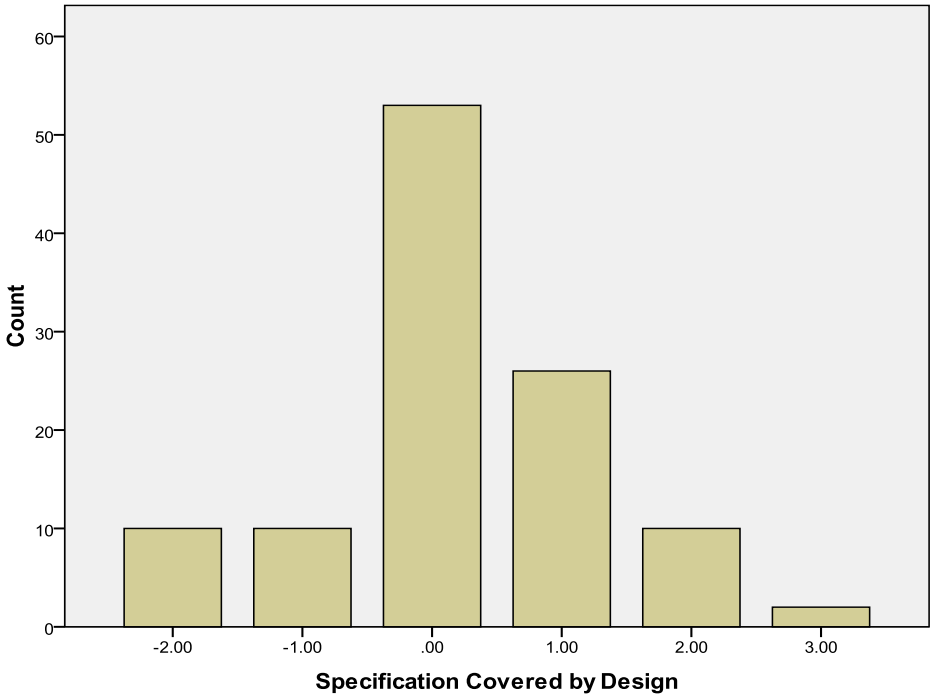


Figure 14: Specification Covered by Design

As illustrated by the bar chart (Figure 24) the calculated gap between Agile and Waterfall for the above stated question number 7 varies among -2 and +3. This indicates that the positive difference is strong than the negative difference. This means there are people who thoroughly believe that Agile is better than Waterfall in covering the specification in the design.

As can be seen at a glance the highest frequency falls on 0. This means that the most of the respondents assume that there is no difference between the two development methods for the above considered factor. According to the figures presented by the above bar chart 53 entries falls on 0 and the total count falls at the negative side is 20, whereas the total count falls at the positive side is 38.

Question 8 - Our system implementation cover the system design 100 %

Here also the differences were calculated by subtracting weight for Waterfall by the weight for Agile. A positive difference implies that in Agile development system implementation stick on to the system design than in Waterfall and wise versa. The difference equals to zero implies that the respondents have provided the same rating for both methods and there is no significant difference between the two techniques.

Table 12: Covering the System Design in the Implementation

Q8 – Implementation 100 % tally with the Design

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	8	7.1	7.3	7.3
	-1.00	12	10.6	11.0	18.3
	.00	49	43.4	45.0	63.3
	1.00	26	23.0	23.9	87.2
	2.00	10	8.8	9.2	96.3
	3.00	4	3.5	3.7	100.0
	Total	109	96.5	100.0	
Missing	System	4	3.5		
Total		113	100.0		

As per the values presented by the above table 12, out of total 113 entries 4 entries had missing values, thus only 109 entries considered in this analysis.

According to the figures shown in the table the calculated differences vary within -2 and +3 where we can approximate that the strongest of the positive side is more than the negative side for the above considered factor.

As per the values presented in the above table, 17.7 % responses of the responses in the negative region and 35.3 % in the positive region and the majority 49 % is on the fence.

Question 9 - Our system implementation 100% free from faults

The differences were calculated by subtracting weight for Waterfall by the weight for Agile. A positive difference implies that in Agile development system implementation is free from faults than in Waterfall; where as a negative difference implies the other way round. The difference equals to zero presume that there is no difference between the two development methods.

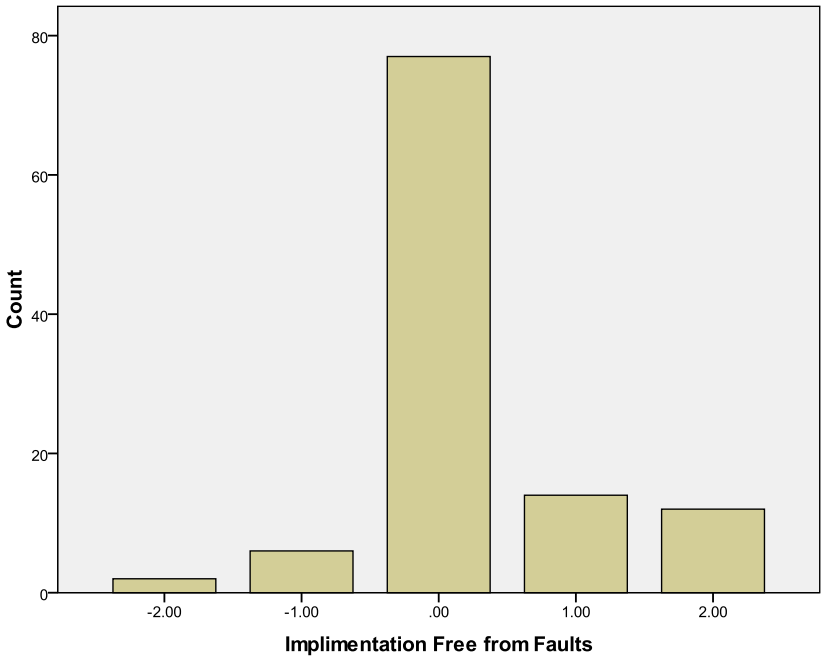


Figure 15: Implementation Free from Faults

According to the figures presented in the above bar chart (Figure 25) the calculated differences varies between -2 and +2. This indicates that the variation does not go beyond ± 2 reflecting that though there is a difference between two methods, the gap between the two methods is small. (For example no one has rated 1 for Agile and 5/4 for Water fall or vice versa or 2 for Agile and 5for Waterfall or vice versa)

As can be seen at a glance in the above chart the majority 77 of the responses fall on the fence. 26 respondents fall at the positive region and only 8 in the negative region. Out of total deviated responses there are more responses in the positive region. This

means from those who articulated that there is a difference most assume that in Agile development implementation is free from faults than in the Waterfall development.

Hypothesis: 01 – There is no mean difference in Correctness between the two development methods Agile and Waterfall

This hypothesis can be check by following null and alternative hypothesis

$$H_0 - \mu_{Acorrectness} - \mu_{Wcorrectness} = 0$$

$$H_1 - \mu_{Acorrectness} - \mu_{Wcorrectness} \neq 0$$

After analysing the calculated gaps in each related question independently, the average gap of all the related questions was considered. Since the gap has been considered as the data set One sample - T test is used to test the above intended Hypothesis.

Therefore we can modify the above hypothesis by considering the calculated gap for Correctness between Agile and Waterfall as follows.

$$H_0 - \mu_{GapCorrectnes} = 0$$

$$H_1 - \mu_{GapCorrectnes} \neq 0$$

Table 13: One Sample Test for Correctness

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Correctness	1.547	112	.125	.13717	-.0385	.3128

Significant value in the table 13 is greater than 0.05; therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Correctness between the software products developed using Agile and Waterfall techniques which means ‘there is no difference in Correctness between the software products developed using Agile and Waterfall methods’

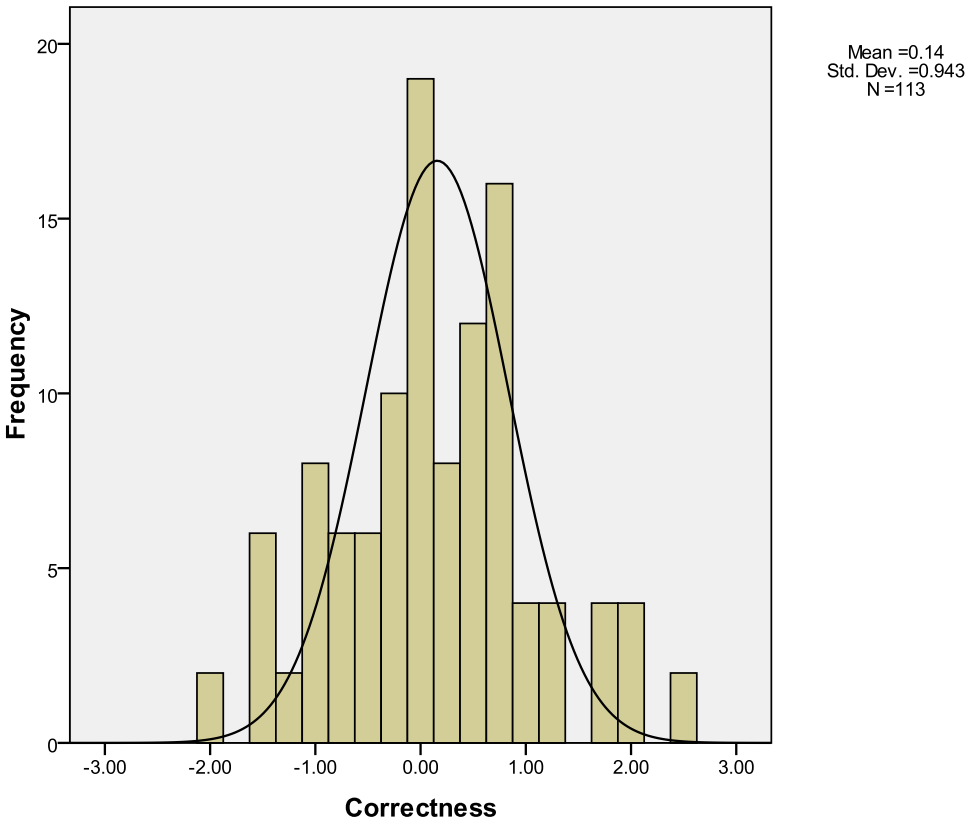


Figure 16: Histogram of Correctness

The histogram above (Figure 26) illustrates that the data are approximately symmetric. There do not appear any significant outliers in the tails. And it seems reasonable to assume that the data are formed in the order of a normal distribution. According to the figures presented in the Histogram: 26 most of the high frequencies are positioned close to zero. Therefore, we can take for granted that there is much likelihood toward our null hypothesis, ‘there is no significant difference in

Correctness of the product between the two development methods of Agile and Waterfall.

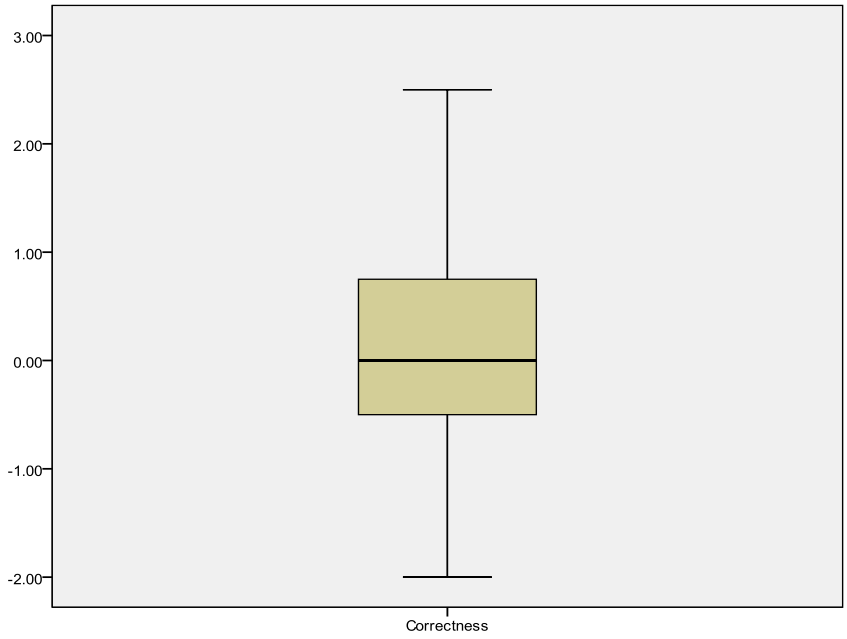


Figure 17: Box plot of Correctness

This can be further justified by the box plot (Figure 27) given above. In the box plot there are no outliers marked. The right whisker is appearing to be equal to the left whisker. This indicates that the distribution is approximately symmetric; thus it is reasonable to assume that the data has normal distribution. Furthermore, the central tendency given by the box plot which is the median is equal to zero. As shown by the histogram (Figure 26) the mean is equal to 0.14 which almost equals to zero. Since the mean and median is equal to zero and the data has a normal distribution we can justify that the mode is also equal to zero.

As per the information presented by the above two figures (Figure 26 & 27) we can justify the suitability of the T-test (Table 13) for the quality factor 'Correctness'

4.3.2 TESTABILITY

As presented in Chapter: 3 Methodology, questions 10 to 12 and 15 in the questionnaire has been designed to capture the information related to the quality factor – 'Testability' This section briefly analyses the responses received for each question and further provide a summary analysis using One sample T – test to verify the derived hypothesis.

Question 10 - We accomplish complete execution of test scripts

The calculation is done by subtracting the weight receives for Waterfall by the weight receives for Agile. Thus a positive difference implies that Agile is better in executing test scripts than Waterfall and a negative difference implies that Waterfall is better in executing test scripts than Agile. Difference equals 0 can results when the rating is same for both methods and implies that the respondent perceives that there is no significant difference between the two methods.

Table 14: Execution of the Test Scripts

Execution of test scripts

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	4	3.5	3.5	3.5
	-1.00	10	8.8	8.8	12.4
	.00	75	66.4	66.4	78.8
	1.00	16	14.2	14.2	92.9
	2.00	8	7.1	7.1	100.0
	Total	113	100.0	100.0	

As can be seen in Table 14 the calculated differences between Agile and Waterfall for question number 10 vary between -2 and +2. Since the variation does not go beyond ± 2 we can conclude that though there is a variation between Agile and Waterfall in

relation to completion of executing test scripts, no respondent perceives that there is a huge difference between the two development methods. (For example no one has rated 1 for Agile and 5 for Water fall or wise versa)

According to the values presented in table 14, out of 113 responses received, 12.3 % of responses plunge at negative region and 21.3 % plunge at positive region. The majority 66.4 % falls on 0 where the perception is towards no difference between the two methods. Since count at positive side is more than the count at negative side, out of those who perceive that there is a difference, more believes that Agile is better than Waterfall in completing the execution of test scripts.

Question 11 - We always adheres to the Coding standards in implementing our systems.

The analysis is made based on the calculation done by subtracting the weight receives for Waterfall by the weight receives for Agile. Hence, a positive difference implies that in Agile development system implementation adheres to the cording standards than in the Waterfall development. A negative difference implies the other way round. The difference equals to 0 results if the rating is same for both the methods and it implies that there is no difference between the two development methods.

Table 15: System Implementation Adhere to the Coding Standards

System Implementation adhere to Coding Standards

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-1.00	12	10.6	10.6	10.6
	.00	77	68.1	68.1	78.8
	1.00	14	12.4	12.4	91.2
	2.00	8	7.1	7.1	98.2
	4.00	2	1.8	1.8	100.0
	Total	113	100.0	100.0	

As can be seen by the above table 15, the calculated differences between Agile and Waterfall techniques vary between -1 and +4. This describes that there is a huge positive difference compared to the negative difference. (For example one can rate 5 for Agile and 1 for waterfall, but no one has rated 5 for Waterfall and 1, 2, or 3 for Agile)

By looking at the values depicted in the above Table 15 it is reasonable to wrap up that the majority 68.1 % of the respondents assume that there is no difference between the two methods and 21.3 % in the positive region and 10.6 % in the negative region.

Question 12 - We do not employee many complex structures in our codes

Here also the differences are calculated by subtracting Waterfall by Agile as mentioned above.

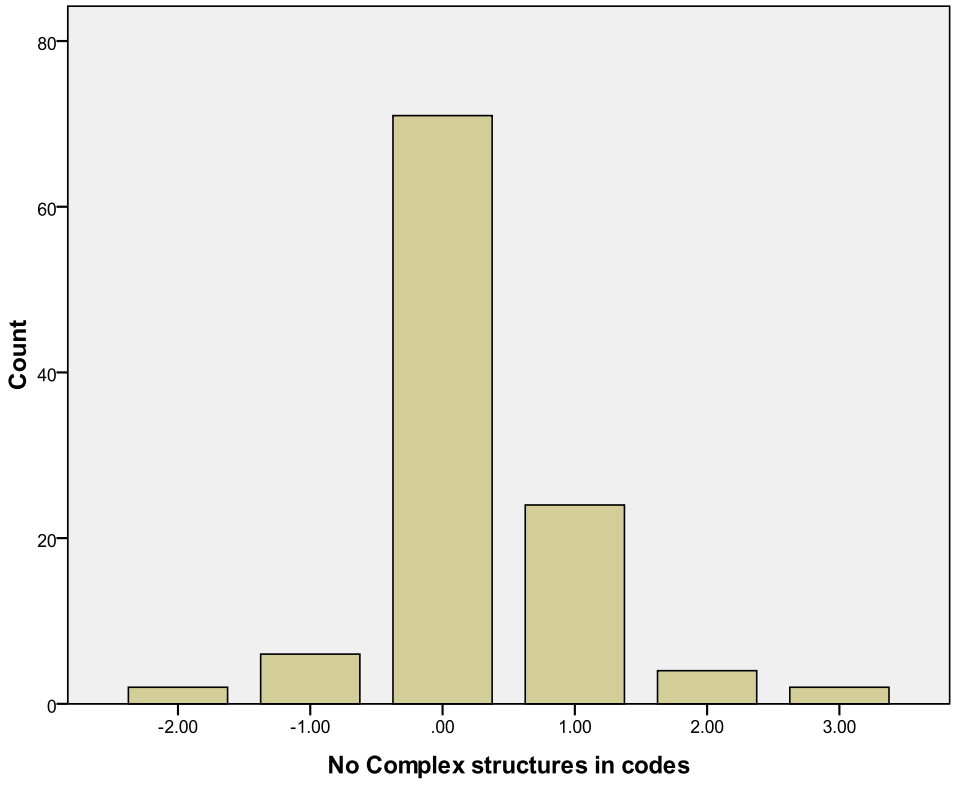


Figure 18: Simple Code Structures

According to the figures presented in the above bar chart (Figure 28) the calculated differences varies between -2 and +3. This means that the positive difference is stronger than the negative difference.

In relation to the frequencies shown in the above bar chart (Figure 28) the majority of the responses fall on 0. As represented by the chunk there are 71 respondents who have provided the same rating for both methods. In the positive region there are 30 respondents and in the negative region there are only 8 respondents. From among those who consider that there is a difference between the two methods most assumes Agile is better than Waterfall in the use of simple coding structures.

Question 15 - Our systems maintain law interaction between modules

For this question also the difference is calculated as described above. Thus, a positive difference indicates that in Agile development there is low interaction between modules compared to Waterfall. A negative difference indicates that the interaction between modules is low in Waterfall development than in Agile

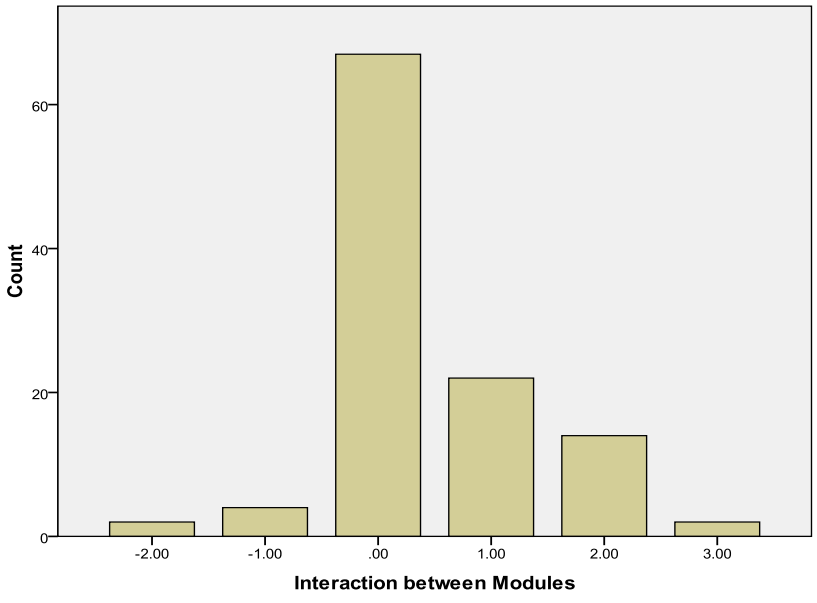


Figure 19: Interaction between Modules

As been figured out by the above bar chart (Figure 29) the calculated differences for Agile and Waterfall techniques for the factor is identified by question number 15 varies between -2 and +3. By analyzing this variance we can sum up that the positive difference is stronger than the negative difference. This means some respondents strongly believe that in Agile there is low interaction between modules than Waterfall development.

By looking at the frequencies depicted by the bars we can see that the highest frequency 67 falls on 0. This means many respondents have provided the same rating for both the methods assuming that there is no significant difference between the two methods. On the other hand, there are 38 responses in the positive region and the minority 6 responses are in the negative region.

Hypothesis 2 - There is no mean difference in Testability between the two development methods Agile and Waterfall

This hypothesis can be check by following null and alternative hypothesis

$$\mathbf{H_0 - \mu_{Atestability} - \mu_{Wtestability} = 0}$$

$$\mathbf{H_1 - \mu_{Atestability} - \mu_{Wtestability} \neq 0}$$

After analysing the calculated gaps in each related question independently, the average gap of all the related questions has been considered. Since the gap has been considered as the data set One sample - T test is used to test the above intended Hypothesis.

Therefore, we can modify the above hypothesis by considering the calculated gap for Testability between Agile and Waterfall as follows:

$$H_0 - \mu_{GapTestability} = 0$$

$$H_1 - \mu_{GapTestability} \neq 0$$

Table 16: One Sample Test for Testability

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Testability	4.414	112	.000	.15929	.0878	.2308

Significant value in the Table 16 is less than 0.05; therefore; we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no difference in Testability between the software products developed using Agile and Waterfall techniques. This analysis proven that

As presented in Table 16 the confidence interval of difference falls between .0878 and .2308. Since 0 does not fall within the interval and also when both limits are positive we can conclude that the differences are mostly positive

As discussed above, the differences are calculated by subtracting weight for Waterfall by the weight for Agile and the questionnaire consisted of positive side questions where a positive difference implies that Agile is better than Waterfall; where as a negative difference implies that Waterfall is better for the considered quality factor. As per the values represented in the above table (Table 16), since we have a positive difference we can wrap up that the Agile is more testable than Waterfall.

Finally, we can conclude that there is a difference between Agile and Waterfall for the quality factor Testability and the software products developed using Agile are more testable than the products developed using Waterfall; hence Agile is at the forefront.

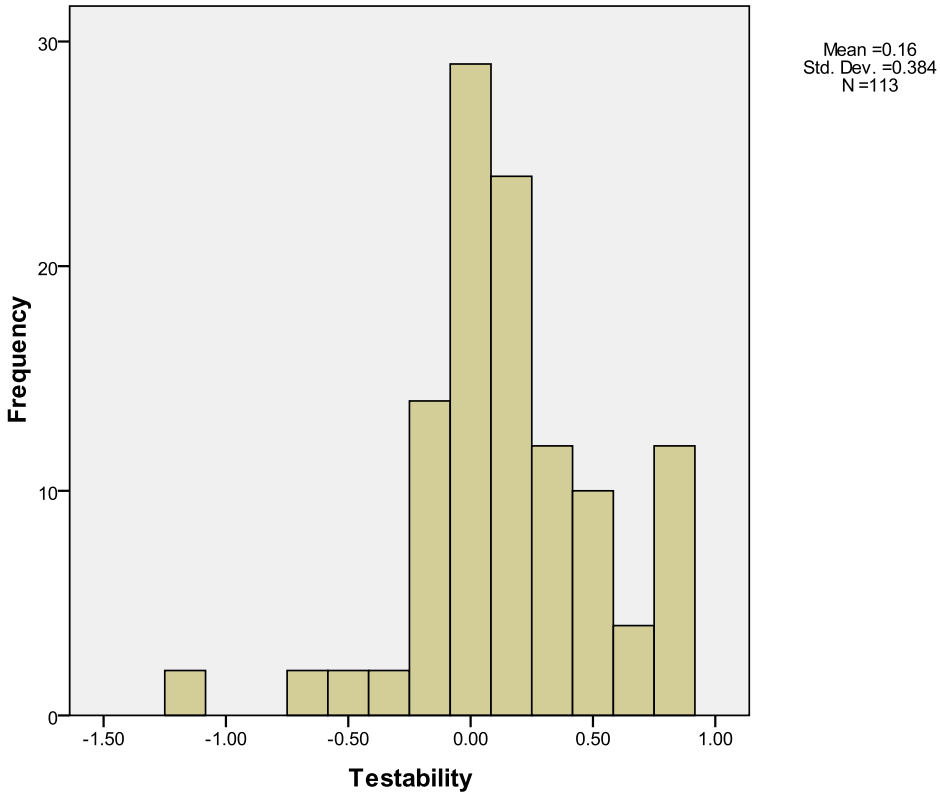


Figure 20: Histogram of Testability

The histogram above (Figure 30) illustrates that the data are approximately left skewed. A significant outlier appears in the left tail. Therefore, it seems reasonable to assume that the data are form in the order of a skewed distribution. According to the figures presented in the histogram (Figure 30) most of the high frequencies are positioned at the positive region (right hand side). Thus we can take for granted that there is much likelihood toward the decision that Agile is better than Waterfall for the considered quality factor Testability.

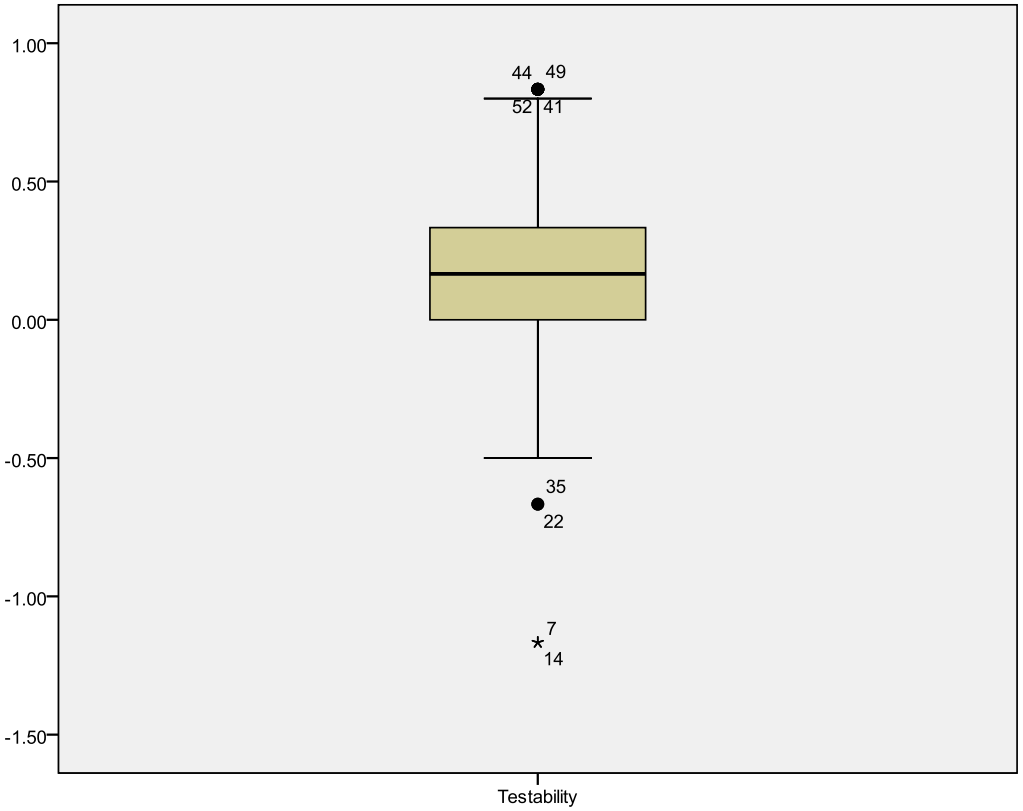


Figure 21: Box Plot of Testability

This can be further justify by the box plot (Figure 31) given above. According to the data presented in the box plot there are two outliers on either side and two extreme outliers in the negative region. As can be seen by the diagram, 2^{nd}, 3^{rd} and the 4^{th} quartiles are located above the zero. This means that the 75 % of the data are positioned in the positive region, where as only 25 % of data falls at the negative side. Hence, we can conclude that there is much likelihood toward Agile development than Waterfall method for the quality factor Testability.

As per the information presented by the above two figures (Figure 30 & 32) we can justify the suitability of the T-test (Table 16) for the quality factor 'Testability'

4.3.3 CHANGEABILITY

As presented in the chapter on methodology, questions 13 to 17 in the questionnaire have been designed to capture the information related to the quality factor – 'Changeability' This section briefly analyses the responses received for each question and further provide a summary analysis using One sample T – test to verify the derived hypothesis.

Question 13 - Modifications can be done to our products without much difficulty

To analyze the responses received for this question, the calculation is done by subtracting the weight receives for Waterfall by the weight receives for Agile. Thus, a positive difference implies that in Agile development modifications to a product is easier than in the Waterfall development. A negative difference implies the modifications are easy in Waterfall method. The difference equals to 0 means that there is no difference in the two methods in modifying the products.

Table 17: Ease of Modifying the Products

Ease of Modifying the products

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	6	5.3	5.3	5.3
	-1.00	4	3.5	3.5	8.8
	.00	57	50.4	50.4	59.3
	1.00	30	26.5	26.5	85.8
	2.00	14	12.4	12.4	98.2
	3.00	2	1.8	1.8	100.0
	Total	113	100.0	100.0	

As can be seen by the above Table 17 the calculated differences between Agile and Waterfall techniques vary between -2 and +3. This describes that positive difference is stronger than the negative difference.

By looking at the values depicted in the above table (Table: 17) it is reasonable to conclude that the majority 50.4 % of the respondents assume that there is no difference between the two methods. And 40.7 % in the positive region suppose that Agile techniques are better in incorporating modifications than Waterfall and the minority 8.8 % in the negative region assuming that Waterfall is healthier.

Question 14 - Our systems do not need much effort to accommodate minor specification changes

Similar to above, Question 13 the differences were calculated by subtracting Waterfall by Agile as well.

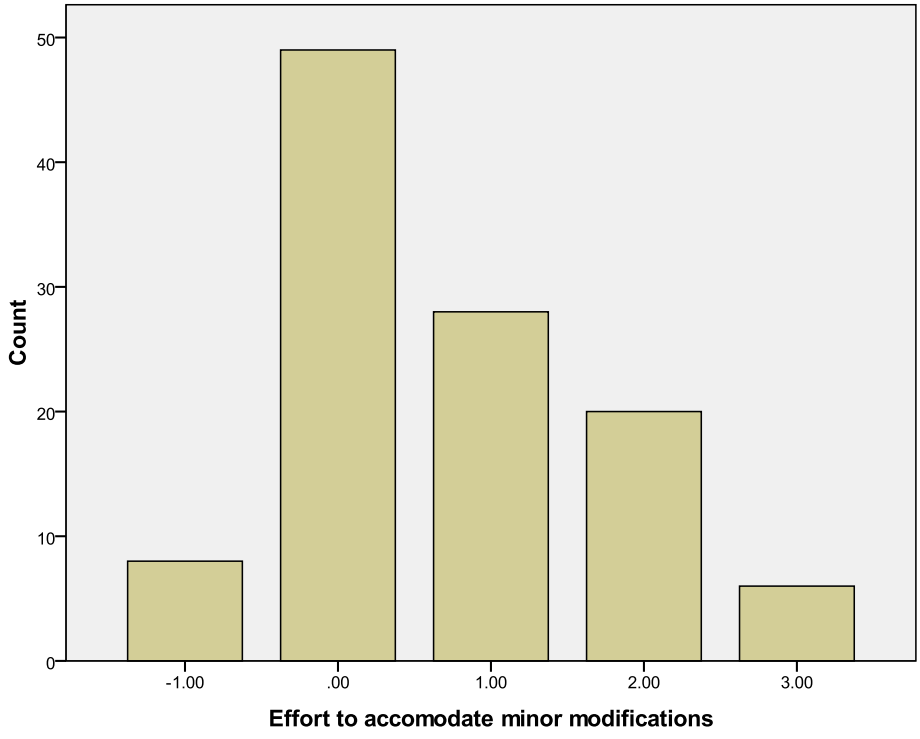


Figure 22: Effort to Accommodate Minor modifications

As been figured out from the bar chart (figure 32) above, the calculated differences for Agile and Waterfall techniques for the factor identified by question number 14 varies between -2 and +3. By analyzing this variance we can sum up that the positive difference is stronger than the negative difference. This means some respondents

strongly believe that Agile is better than Waterfall in accommodating specification changes.

By looking at the frequencies depicted by the bars we can see that the highest number of responses falls at the positive region and the lowest number of responses at the negative region. And a moderate amount falls on 0. When considering the amounts, 54 in positive side, 8 in the negative side and 49 on zero. Further, analyzing these values it is reasonable to settle on that more of the respondents assume that Agile is better than the Waterfall when accommodating specification changes.

Question 15 - Our systems maintain law interaction between modules

The calculation is done by subtracting the weight receives for Waterfall by the weight receives for Agile. A positive difference implies that interaction between modules is less in Agile, where as a negative difference implies the interaction between modules is low in Waterfall method. The calculated value equals to 0 means that there is no difference in two methods for the above considered factor.

Table 18: Interaction between Modules

Interaction between Modules

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	2	1.8	1.8	1.8
	-1.00	4	3.5	3.6	5.4
	.00	67	59.3	60.4	65.8
	1.00	22	19.5	19.8	85.6
	2.00	14	12.4	12.6	98.2
	3.00	2	1.8	1.8	100.0
	Total	111	98.2	100.0	
Missing	System	2	1.8		
Total		113	100.0		

As per the figures in the above table there are 2 missing values therefore out of 113 entries only 111 responses has been considered for the analysis.

According to the values presented in the above table 18, the calculated differences between Agile and Waterfall techniques vary between -2 and +3. This describes that positive difference is stronger than the negative difference.

By looking at the values depicted in the above Table 4.13, it is reasonable to conclude that the majority 59.3 % of the respondents have provided the same ratings for both the methods assume that there is no difference between the two methods. 33.7 % respondents in the positive region suppose that the interaction between modules is low in Agile development compared to Waterfall. The minority 6 % in the negative region assume that Waterfall is better in achieving low interaction between modules.

Question 16 - When the changes are done to one module our systems has very low side effects to other modules

Similar to previous questions here also the analysis is done by subtracting the rating for Waterfall by the rating for Agile.

As been figured out from the above bar chart (figure 33) the calculated differences for Agile and Waterfall techniques for the factor identified by question number 16 varies between -4 and +3. By analyzing this variance we can sum up that the negative difference is stronger than the positive difference. This means some respondents strongly believe that in Waterfall method changes to a module can be done without much affecting the other modules when compared to Agile techniques.

By looking at the frequencies depicted by the bars we can see that the highest number of responses falls on 0. A moderate amount falls at the positive side and a little amount in the negative side. Further analyzing these values it is reasonable to reconcile that more of the respondents assume that there is no difference between two methods and though very few strongly believe that Waterfall is better, many think that with Agile the side effects of modifications is low.

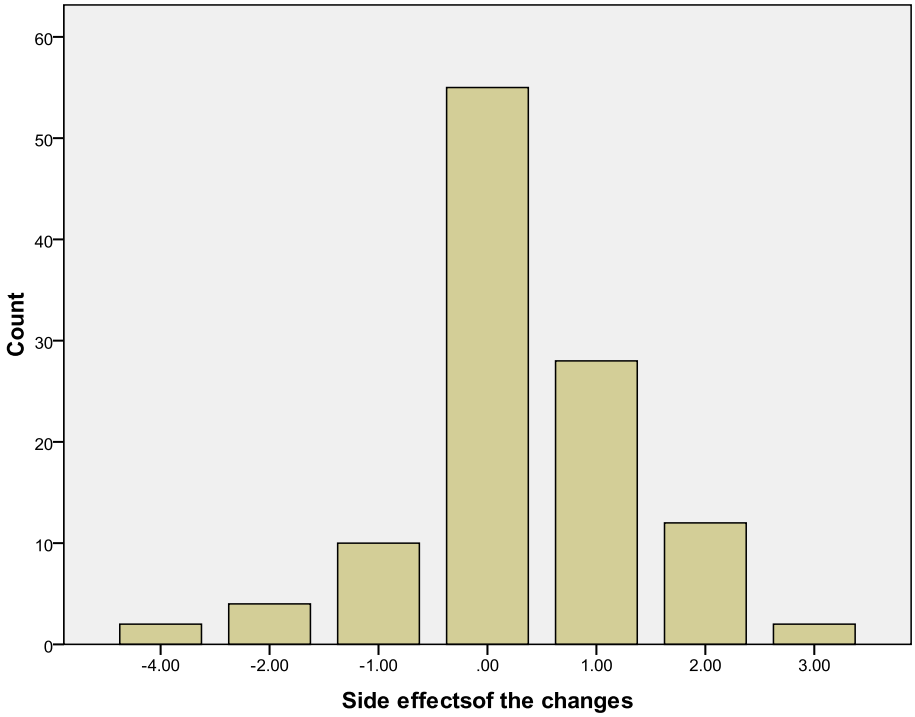


Figure 23: Side effects of the Changes

Question 17 - Integration of a new component to the system does not create many functional issues

The calculation is done by subtracting the weights receive for Waterfall by the weights received for Agile. Therefore, positive difference implies that there are very low functional issues in Agile development when integrating new components. Where as a negative difference signifies that the Waterfall is better in integrating new components. And the calculated value equals to zero means that there is no difference between the two development methods.

Table 19: Ease of integrating new components

Ease of integrating new components

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	4	3.5	3.5	3.5
	-1.00	6	5.3	5.3	8.8
	.00	69	61.1	61.1	69.9
	1.00	14	12.4	12.4	82.3
	2.00	16	14.2	14.2	96.5
	3.00	4	3.5	3.5	100.0
	Total	113	100.0	100.0	

According to the figures presented in the above table 19 there are no missing values; therefore, total 113 responses entered is considered in this study.

As per the data presented in the table 19, the calculated differences between Agile and Waterfall for the question number 17 varies between -2 and +3. Therefore, we can further conclude that the positive side is much stronger than the negative side.

As shown in the above table there are total of 34 respondents falls at the positive region and 10 respondents falls at the negative region, where as majority 69 responses lays where the calculated value is equals to zero.

Therefore by analyzing the totals depicted by Table 19, the majority of respondents believe that there is no difference between the two techniques in integrating new components. When considering the positive and negative regions more respondents fall at the positive region; whereas less count in the negative region. This means that few respondents believe that compared to Agile, Waterfall makes less functional issues when integrating new components.

Hypothesis 3 - There is no mean difference in Changeability between the two development methods Agile and Waterfall

This hypothesis can be verified by following null and alternative hypothesis

$$H_0 - \mu_{Achangeability} - \mu_{Wchangeability} = 0$$

$$H_1 - \mu_{Achangeability} - \mu_{Wchangeability} \neq 0$$

After analysing the calculated gaps in each related question independently, the average gap of all the related questions has been considered. Since the gap has been considered as the data set One sample - T test is used to test the above intended Hypothesis.

Therefore, we can modify the above hypothesis by considering the calculated gap for Changeability between Agile and Waterfall as follows.

$$H_0 - \mu_{GapChangeability} = 0$$

$$H_1 - \mu_{GapChangeability} \neq 0$$

Table 20: One Sample Test for Changeability

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Changeability	6.309	112	.000	.44513	.3053	.5849

Significant value in the table 20 is less than 0.05; therefore, we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to show that there is no difference in Changeability between the software products developed using Agile and Waterfall techniques; which means ‘there is a difference in

Changeability between the software products developed using Agile and Waterfall methods'

In table 20 the confidence interval of the differences falls between .3053 and .5849. Since 0 does not fall within the interval and also both the limits are positive we can conclude that the differences are mostly positive.

As discussed above the differences are calculated by subtracting Waterfall by Agile and the questionnaire consisted of positive side questions. A positive difference implies that the products develop using Agile is more changeable that the products developed using Waterfall; where as a negative difference implies that the products in Waterfall is more changeable. Since we have a positive difference as per the analysis above, we can wrap up that for the quality factor 'Changeability' Agile is better than Waterfall.

Finally, we can conclude that there is a difference between Agile and Waterfall for the quality factor changeability and the software products developed using Agile are more changeable than the products developed using Waterfall; hence Agile is in the forefront.

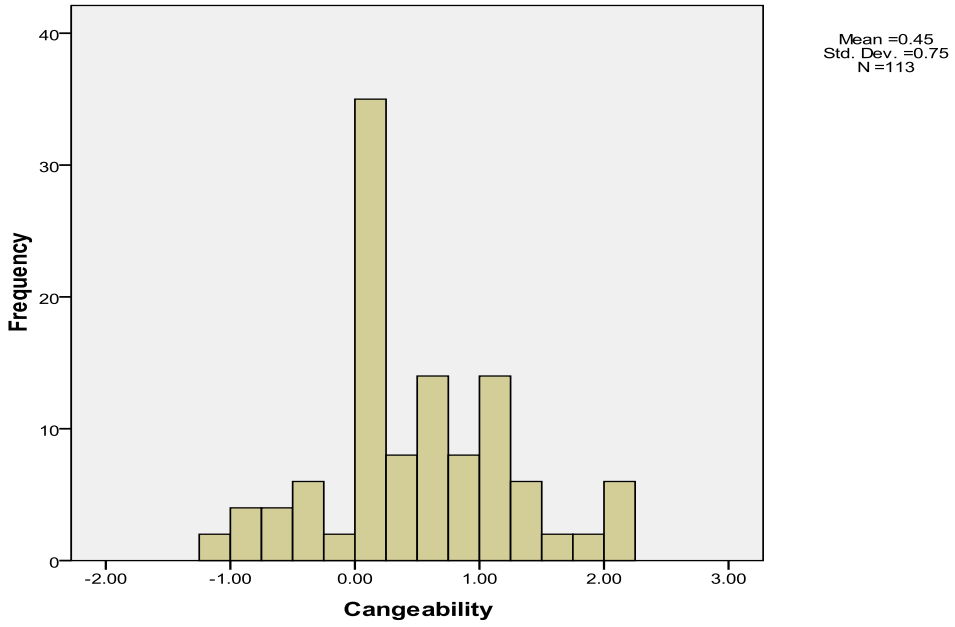


Figure 24: Histogram of Changeability

The histogram above (Figure 34) illustrates that the data are approximately left skewed. There are no significant outliers in the diagram. Also it seems reasonable to assume that the data are formed in the order of skewed distribution and are left skewed. According to the figures presented in the histogram (Figure 34) most of the high frequencies are positioned at the positive region (right hand side). Thus, we can take for granted that there is much likelihood toward the decision that Agile is better than Waterfall for the considered quality factor Changeability.

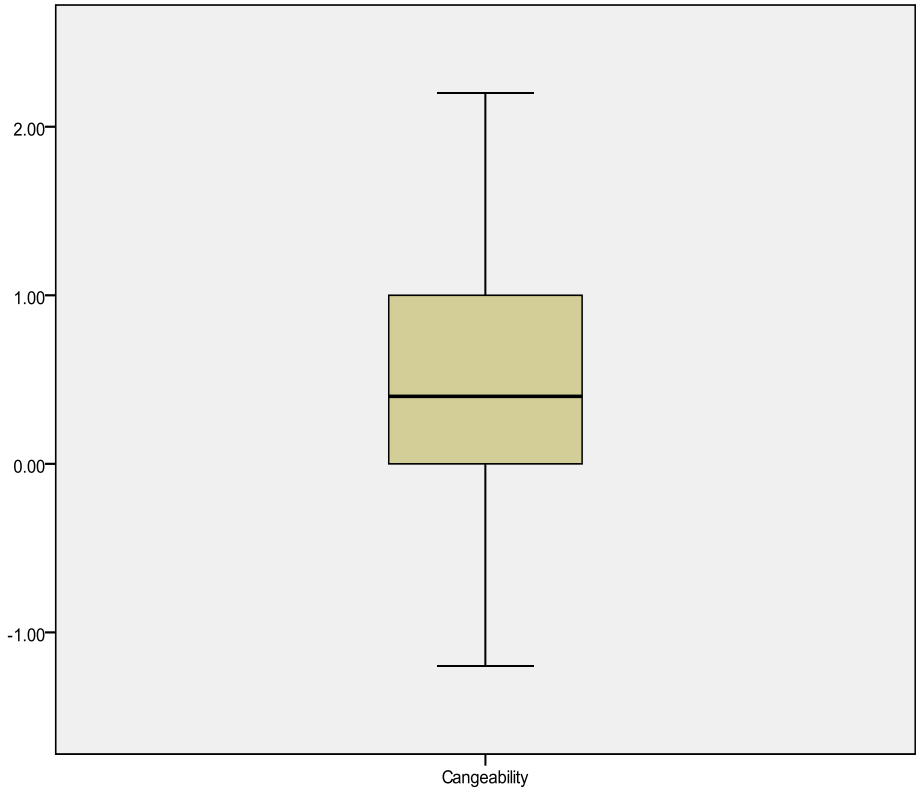


Figure 25: Box Plot of Changeability

This can be further justified by the box plot (Figure 35) given above. According to the data presented in the box plot there are no significant outliers in the two tails. As clearly depicted by the above diagram (Figure 35) 2nd, 3rd and the 4th quartiles are located above zero. This means that the 75 % of the data are positioned in the positive region, where as only 25 % of data falls at the negative side. Hence, we can conclude that there is much likelihood toward Agile development than Waterfall method.

As per the information presented by the above two figures (Figure 34 & 35) we can justify the suitability of the T-test (Table 20) for the quality factor 'Changeability`

4.3.4 INSTALL ABILITY

As presented in Chapter 3: Methodology, questions 18 to 20 in the questionnaire has been designed to capture the information related to the quality factor – 'Install ability` This section briefly analyses the responses received for each question and further provides a summary analysis using One sample T – test to verify the derived hypothesis.

Question 18 - Our systems do not challenge during the installation in the agreed environment

The calculation is done by subtracting the weight receives for Waterfall by the weight receives for Agile. A positive difference implies that compared to Waterfall there is less challenges during the installation in Agile development. And a negative difference signifies that Waterfall is better than Agile.

Table 21: No Changers in Installation

No challenges in the installation

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-2.00	2	1.8	1.8	1.8
	-1.00	12	10.6	10.8	12.6
	.00	77	68.1	69.4	82.0
	1.00	14	12.4	12.6	94.6
	2.00	6	5.3	5.4	100.0
	Total	111	98.2	100.0	
Missing	System	2	1.8		
Total		113	100.0		

According to the values presented in Table 21, the calculated difference between Agile and Waterfall for level of challenge during the installation varies between -2 and +2. Since the variation does not go beyond more than ± 2 we can conclude that though there is a variation between Agile and Waterfall, no respondent perceive that there is a huge difference between the two development methods. (For example no one has rated 1 for Agile and 5 for Water fall or wise versa)

According to the values presented in table 21, out of 113 responses received, 12.6 % of responses fall at the negative region; where as 17.7 % falls at the positive region. The majority 68.1 % falls on 0. This reflects the fact that most of the respondents assume that there is no difference between the two methods for the level of challengers meets at the installation.

Question 19 - We have to incorporate minor modifications when installing our systems in the agreed environment

Here also the differences were calculated by subtracting the weight receive Waterfall by the weight receive for Agile

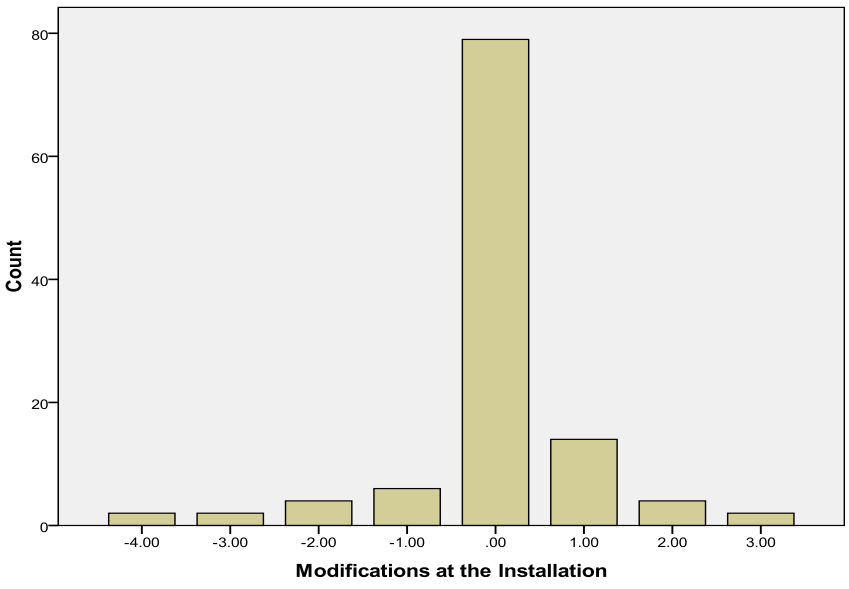


Figure 26: Modifications at the Installation

As been figured out by the bar chart above (Figure 36), the calculated differences for Agile and Waterfall vary between -4 and +3. According to the variation we can suggest that the negative difference is stronger than the positive difference. As clearly depicted by the graph the responses fall on zero are dramatically large compared to the values in the negative and the positive sides. According to the figures presented in the chart there are only 14 respondents in the negative side, where as 20 falls at the positive side and the majority 79 responses are on the fence.

Question 20 - Hardware configuration is always compatible with software we developed when deploying our systems

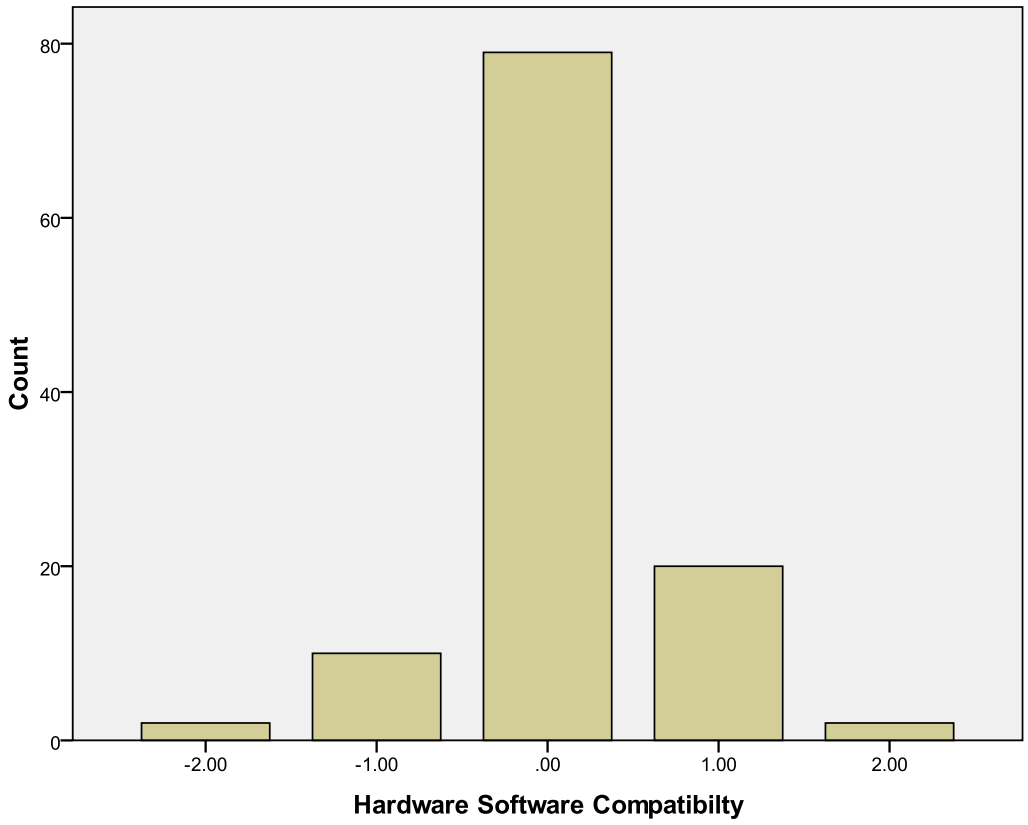


Figure 27: Hardware Software compatibility

According to the values presented in the bar chart above (Figure 37), the calculated differences between Agile and Waterfall for question number 20 varies between -2 and +2. Since the variation does not go beyond ± 2 we can conclude that though there is a variation between Agile and Waterfall in relation to hardware and software compatibility, no respondents have perceived a huge difference between the two development methods. (For example no one has rated 1 for Agile and 5 for Water fall or wise versa)

As been clearly depicted by the graph (Figure 37) the responses fall on zero are dramatically large to the values falls on the negative and the positive sides. According to the figures presented in the chart above there are only 12 respondents in the negative side, where as 22 falls at the positive side and the majority 79 responses is on the fence.

Hypothesis 4- There is no mean difference in Install ability between the two development methods Agile and Waterfall

This hypothesis can be check by following null and alternative hypothesis

$$\mathbf{H_0 - \mu_{Ainstalability} - \mu_{Winstalability} = 0}$$

$$\mathbf{H_1 - \mu_{Ainstalability} - \mu_{Winstalability} \neq 0}$$

After analysing the calculated gaps in each related question independently, the average gap of all the related questions has been measured. Since the gap has been considered as the data set One sample - T test is used to test the above intended Hypothesis.

Therefore, we can modify the above hypothesis by considering the calculated gap for Install ability between Agile and Waterfall methods as follows.

$$H_0 - \mu_{GapChangeability} = 0$$

$$H_1 - \mu_{GapChangeability} \neq 0$$

Table 22: One Sample Test for Install-ability

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Installability	.828	112	.409	.04720	-.0657	.1601

Significant value in the table 22 is greater than 0.05; therefore; we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Install ability between the software products developed using Agile and Waterfall techniques; which means 'there is no difference in Install ability between the software products developed using Agile and Waterfall methods'.

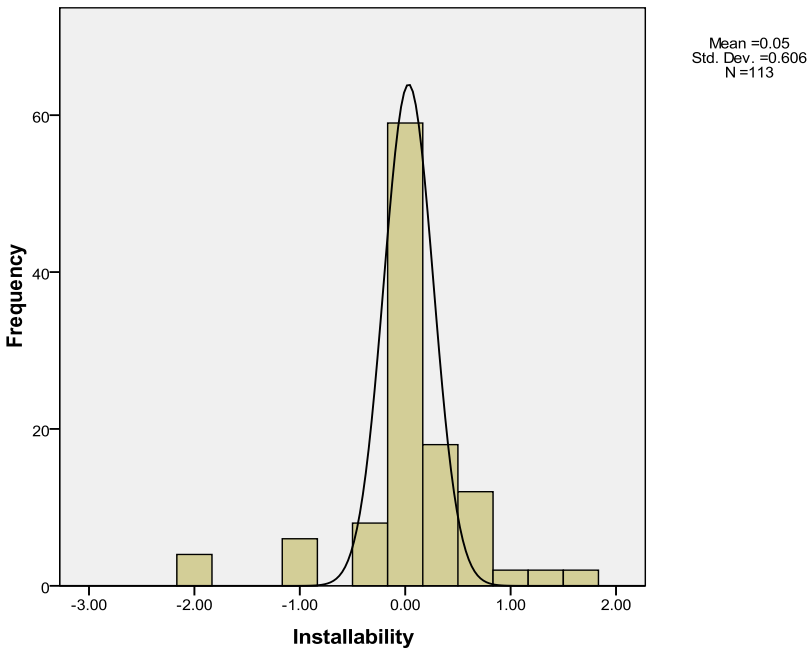


Figure 28: Histogram of Install ability

The histogram above (Figure 38) illustrates that the data are approximately symmetric and there appears to be significant outliers in the left tail. And it seems reasonable to assume that the data are formed in the order of a normal distribution. According to the figures presented in the histogram 38 most of the high frequencies are positioned close to zero; therefore; we can take for granted that there is much likelihood toward that there is no difference between the Agile and Waterfall methods for the above considered quality factor 'Install ability'.

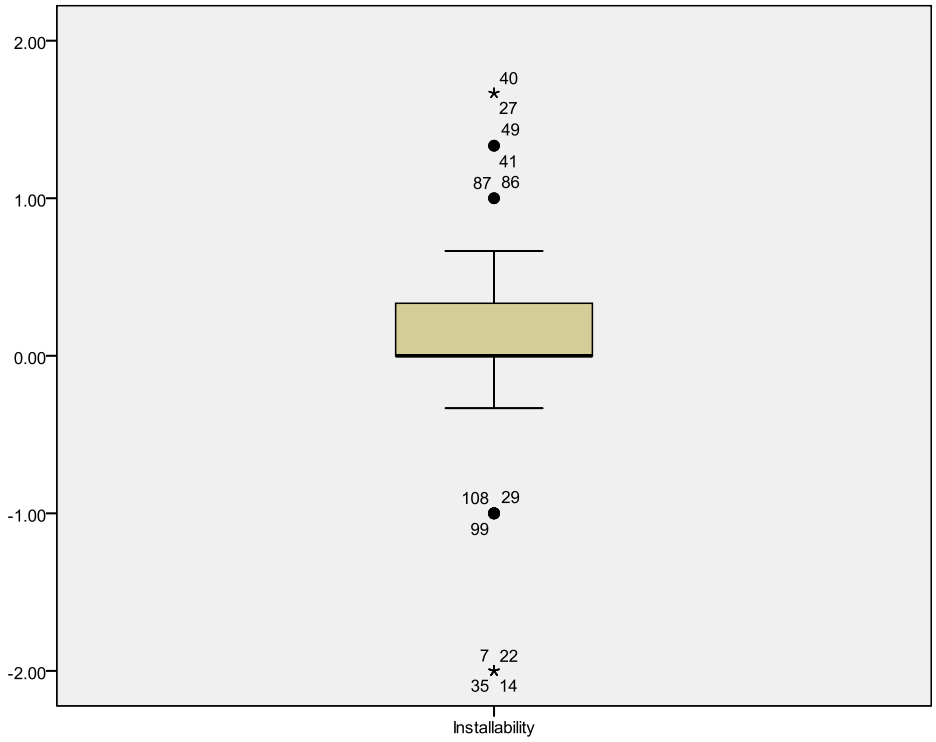


Figure 29: Box Plot of Install ability

This can be further justified by the box plot (Figure 39) given above. As can be seen by the box plot, in the positive region there are 4 outliers and 2 extreme outliers and in the negative region there are 3 outliers and 4 extreme outliers marked. Since the right whisker is appearing to be equal to the left whisker we can guess that the distribution is approximately symmetric with outliers and it is reasonable to assume that the data has normal distribution. Furthermore, the central tendency given by the box plot which is the median equals to zero. As been shown by the histogram (Figure 38) the

mean is equals to 0.05 which almost equals zero. Since the mean and median is equals to zero and the data has a normal distribution we can justify that the mode is also equal to zero.

As per the information presented by the above two figures (Figure 38 & 39) we can justify the suitability of the T-test (Table 22) for the quality factor 'Install ability'

4.3.5 On Time Delivery

As presented in chapter 3 Methodology, the information retrieved from the interview has been used in this analysis. This section briefly analyses the responses received and uses the Chi Square test to verify the derived hypothesis.

Table 23: Projects Completed on Time

Respondent	Agile	Waterfall	A_CompOnTime	W_CompOn time
1	2	2	1	2
2	6	1	6	1
3	7	2	5	1
4	2	3	0	2
5	1	4	1	2
6	1	10	1	6
7	2	8	2	3
8	1	2	1	1
9	3	5	1	2
10	5	9	3	4
11	2	2	1	2
12	7	2	5	0
Total	**39**	**50**	27	26
Percentage			69.23	52

The above table 23 shows the data collected from 12 project managers in relation to the total number of projects completed during the past 5 years. According to the data presented in table there are 39 total projects for Agile and 50 projects for Waterfall. When considering the % of projects completed on time for Agile there are 69.23 %,

where as for Waterfall there are 52 %. By looking at the values presented it is reasonable for us to assume that Agile is better for on time delivery when compared to Waterfall.

Hypothesis 5 – There is no difference in Time to market between the two development techniques.

$$H_0 - \mu_{Atime} - \mu_{Wtime} = 0$$

$$H_1 - \mu_{Atime} - \mu_{Wtime} \neq 0$$

Table 24: Chi Square of Time

Method		Agile	Waterfall	Total	
Successful	Observed	27	26	53	
	Expected	23.22	29.78	53	
Unsuccessful	Observed	12	24	36	
	Expected	15.78	20.22	36	
		39	50	89	
		$\chi 2$ =	3.78	14.25275	0.613689
			-3.78	14.25275	0.478677
			-3.78	14.25275	0.903486
			3.78	14.25275	0.704719
		Test Statistic			2.700571
		Critical value		3.841	

According to the values presented in the above table 24, the calculated Chi Square value for time period is 2.70 and the critical value for the sample is 3.8. Since the observed value is less than the critical value there is no evidence to reject our null hypothesis at 5 % significant level. Therefore, we can conclude that for the above identified quality factor 'On time delivery' statistically there is no significant

difference between the two development methods, though at a glance (Table 23) it seems that Agile is better.

4.3.7 ON BUDGET DELIVERY

As presented in the chapter 3 Methodology, information retrieved from the Project Managers via interview is used in this analysis. This section briefly analyses the responses received and uses the Chi Square test to verify the derived hypothesis.

Table 25: Projects Completed within the Budget

Respondent	Agile	Waterfall	A_CompOnBudget	W_CompOnBudget
1	2	2	1	2
2	6	1	4	0
3	7	2	6	1
4	2	3	0	1
5	1	4	1	3
6	1	10	0	6
7	2	8	1	3
8	1	2	1	1
9	3	5	0	2
10	5	9	2	3
11	2	2	1	2
12	7	2	6	0
Total	**39**	**50**	23	24
Percentage			58.97	48

The above table 25 shows the data collected from 12 project managers in relation to the total number of projects completed during the past 5 years. According to the data presented in the table there are 39 total projects for Agile and 50 projects for Waterfall. When considering the percentages of projects completed within the budget; for Agile there are 58.97 %, where as for Waterfall there are 48 %. By looking at the values presented in the table it is reasonable for us to assume that Agile is better to complete the projects on time than Waterfall.

Hypothesis 6 – There is no difference in the budget between the two development techniques Agile and Waterfall

$$H_0 - \mu_{Abudget} - \mu_{Wbudget} = 0$$

$$H_1 - \mu_{Abudget} - \mu_{Wbudget} \neq 0$$

Table 26: Chi Square of Budget

Method		Agile	Waterfall	Total	
Successful	Observed	23	24	47	
	Expected	20.60	26.40	47	
Unsuccessful	Observed	16	26	42	
	Expected	18.40	23.60	42	
		39	50	89	
		χ2 =	2.40	5.781593	0.280721
			-2.40	5.781593	0.218962
			-2.40	5.781593	0.31414
			2.40	5.781593	0.245029
		Test Statistic			1.058853
		Critical value			3.841

According to the values presented in the above table 26, the calculated Chi Square value for time period is 1.05 and the critical value for the sample is 3.8. Since the observed value is less than the critical value, there is no evidence to reject our null hypothesis at 5 % significant level. Therefore, we can conclude that for the above identified quality factor 'Deliver on Budget', statistically there is no significant difference between the two development methods, though at a glance (Table 25) it seems that Agile is better.

4.3.7 PRODUCT QUALITY

As discussed in the above sections (Section 4.3. - 1, 2, 3, and4) after analyzing each individual quality factor that contributes to the product quality, a cumulative

investigation has been done using One sample T- test for the calculated average to verify the below specified Hypothesis.

Hypothesis 7

$$H_0 - \mu_{Aproductq} - \mu_{Wproductq} = 0$$

$$H_1 - \mu_{Aproductq} - \mu_{Wproductq} \neq 0$$

Since the average gap has been taken as the data set bellow, the hypothesis is derived and tested using One Sample T test

$$H_0 - \mu_{Gapproductq} = 0$$

$$H_1 - \mu_{Gapproductq} \neq 0$$

Table 27: One Sample Statistics for Product Quality

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Product_ Quality	4.215	112	.000	.19720	.1045	.2899

Significant value in the table 27 is less than 0.05; therefore; we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no difference in Software Product quality between the software products developed using Agile and Waterfall techniques; which means 'there is a difference in Software product quality between the software products developed using Agile and Waterfall methods'.

In table 27 the confidence interval of differences falls between .1045 and .2899. Since 0 does not fall within the interval and also both the limits are positive, we can conclude that the differences are mostly positive.

As discussed above the differences were calculated by subtracting Waterfall by Agile and the questionnaire consisted of positive side questions; a positive difference implies that Agile is favorable, where as a negative difference implies that Waterfall is favorable. Therefore, by analyzing the above figures we can conclude that the software product quality is higher in Software products developed using Agile techniques, when compared to the products developed using Waterfall method.

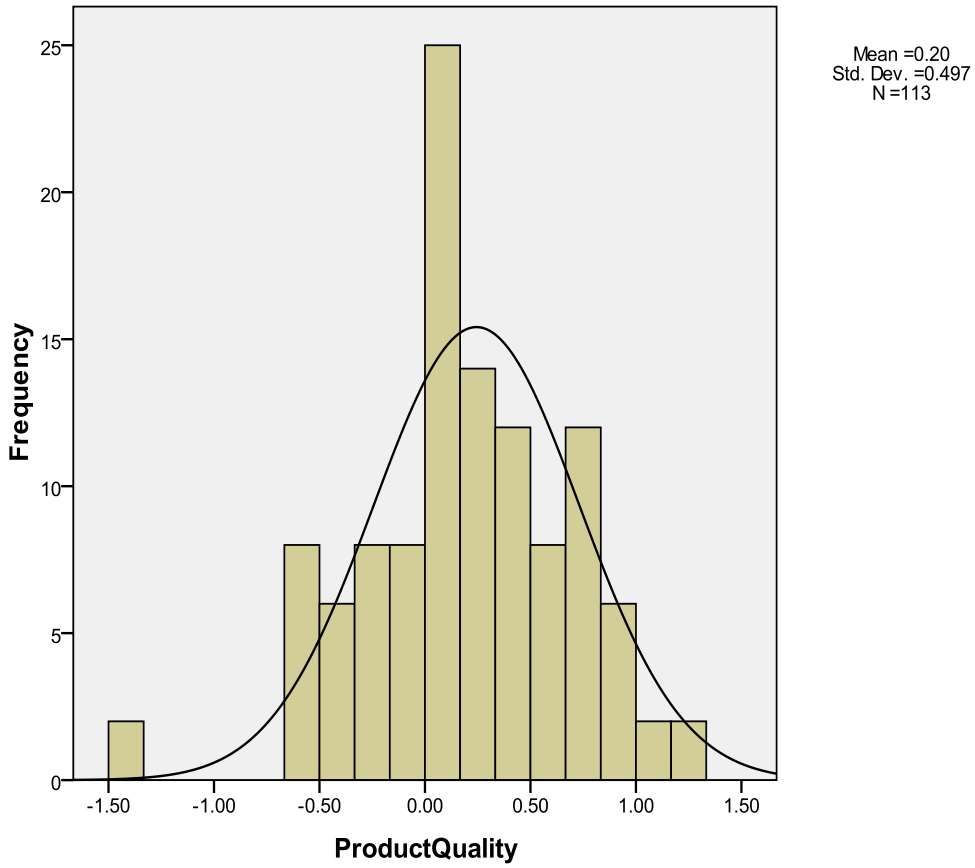


Figure 30: Histogram of Product Quality

The histogram above (Figure 40) illustrates that the data are approximately left skewed and a significant outlier appears in the left tail. And it seems reasonable to assume that the data are formed in the order of a skewed distribution. According to the figures presented in the histogram (Figure 40) most of the high frequencies are

positioned at the positive region (right hand side). Thus, we can take for granted that there is much likelihood toward the decision that Agile is better than Waterfall for the considered factor Product quality.

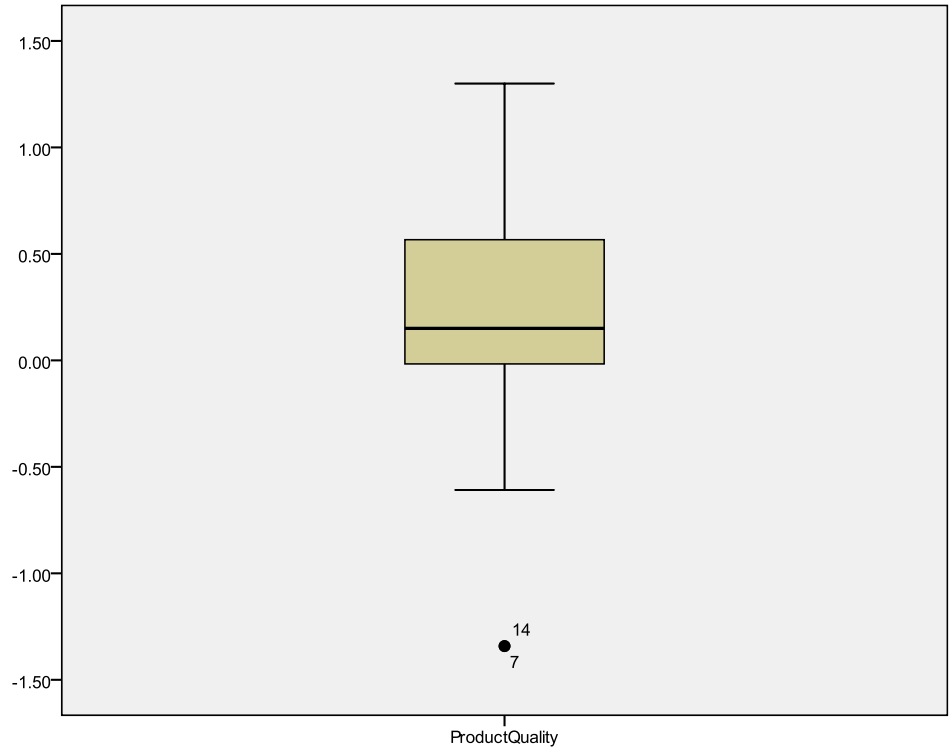


Figure 31: Box Plot of Product Quality

This can be further justified by the box plot (Figure 41) given above. According to the data presented in the box plot there are significant outliers in the left tail. As clearly depicted by the above diagram (Figure 41) 2nd, 3rd and the 4th quartiles are located above zero. This means that 75 % of the data are positioned in the positive region; whereas only 25 % of data falls at the negative side. Hence, we can conclude that there is much likelihood toward Agile development than Waterfall method.

As per the information presented by the above two figures (Figure 40 & 41) we can justify the suitability of the T-test (Table 27) for the factor ‘Product Quality`

4.4 DEMOGRAPHIC ANALYSIS

The demographic analysis below is based on the data collected from 76 Developers, 26 testers and 11 QA leads. The analysis was done against the quality factors 'Correctness', 'Testability', and 'Changeability' and 'Installability' respectively.

4.4.1. CORRECTNESS

Developer

$$H_0 - \mu_{Gap\_Developer\_Correctness} = 0$$

$$H_1 - \mu_{Gap\_Developer\_Correctness} \neq 0$$

The table below presents the analysis done on the quality factor 'Correctness'. And it is based on the responses of 76 developers who participated in the research.

Table 28: One Sample Statistics for Correctness - Developer

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Correctness	2.667	75	.009	.29605	.0750	.5172

Significant value in the table 28 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Correctness between the software products developed using Agile and Waterfall techniques which means from the perspective of the developers 'there is no significant difference in Correctness between the software products developed using Agile and Waterfall methods'

Tester

$$H_0 - \mu_{Gap\_Tester\_Correctness} = 0$$

$$H_1 - \mu_{Gap\_Tester\_Correctness} \neq 0$$

The table below summarises the responses of 26 testers in the sample on the quality factor 'Correctness'

Table 29: One Sample Statistics for Correctness - Tester

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Correctness	-.948	25	.352	-.15385	-.4882	.1805

Significant value in the table 29 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Correctness between the software products developed using Agile and Waterfall techniques which means from the perspective of the testers 'there is no significant difference in Correctness between the software products developed using Agile and Waterfall methods'

QA Lead

$$H_0 - \mu_{Gap\_Developer\_Correctness} = 0$$

$$H_1 - \mu_{Gap\_Developer\_Correctness} \neq 0$$

The table 30 below represents the analysis done on the 11 responses received from QA Leads related to the quality factor 'Correctness'.

Table 30: One Sample Statistics for Correctness – QA Lead

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Correctness	-1.150	10	.277	-.27273	-.8009	.2555

Significant value in the table 30 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Correctness between the software products developed using Agile and Waterfall techniques which means from the perspective of the QA Leads participated in this research 'there is no significant difference in Correctness between the software products developed using Agile and Waterfall methods'

As per the analysis none of the respondent categories believe that there is difference in 'Correctness' between the software products developed using Agile and Waterfall techniques. This demographic analysis further justifies the finding 'There is no significant difference between Agile and Waterfall techniques for the quality factor Correctness' in section 4.2.1 of the document.

4.4.2 TESTABILITY

Developer

The table below presents the analysis done on the developer's perception for quality factor testability between Agile and Waterfall techniques.

$$H_0 - \mu_{Gap\_Developer\_Testability} = 0$$

$$H_1 - \mu_{Gap\_Developer\_Testability} \neq 0$$

Significant value in the Table 31 is less than 0.05 therefore we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no significant difference in Testability between the software products developed using Agile and Waterfall techniques which means from the viewpoint of the developers 'there is a significant difference in Testability between the software products developed using Agile and Waterfall methods'

Table 31: One Sample Statistics for Testability – Developer

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Testability	3.700	75	.000	.17456	.0806	.2685

According to the values presented in the Table 31 the confidence interval of difference falls between .0806 and .2685. Since 0 does not fall within the interval and also both the limits are positive we can conclude that the differences are mostly positive. Since the differences are calculated by subtracting weight receive for Waterfall by the weight for Agile a positive difference implies Agile is better than Waterfall. Therefore we can conclude that from the perception of the developers software products developed using Agile are more testable than the products developed Waterfall method.

Tester

The table below summarizes results of the analysis done on Testability on the view point of testers

$$H_0 - \mu_{Gap\_Tester\_Testability} = 0$$

$$H_1 - \mu_{Gap\_Tester\_Testability} \neq 0$$

Table 32: One Sample Statistics for Testability – Tester

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Testability	2.791	25	.010	.15641	.0410	.2718

Significant value in the Table 32 is less than 0.05 therefore we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no significant difference in Testability between the software products developed using Agile and Waterfall techniques which means from the viewpoint of the developers 'there is a significant difference in Testability between the software products developed using Agile and Waterfall methods'

According to the values presented in the Table 32 the confidence interval of difference falls between .0410 and .2718. Since 0 does not fall within the interval and also both the limits are positive we can conclude that the differences are mostly positive. Since the differences are calculated by subtracting weight for Waterfall by the weight for Agile a positive difference implies Agile is better than Waterfall. Therefore we can conclude that from the perception of the testers software products develop using Agile is more testable than the products develop in Waterfall method.

QA Lead

$$H_0 - \mu_{Gap\_QALead\_Testability} = 0$$

$$H_1 - \mu_{Gap\_QALead\_Testability} \neq 0$$

Table 33: One Sample Statistics for Testability – QA Lead

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Testability	.498	10	.629	.06061	-.2103	.3315

Significant value in the table 33 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Testability between the software products developed using Agile and Waterfall techniques which means from the perspective of the QA Leads participated in this research 'there is no significant difference in Testability between the software products developed using Agile and Waterfall methods'

As per the analysis, Developers and Testers believe that software products developed using Agile are more testable than the products developed using Waterfall. But it reflected that from the view point of QA leads there is no significant difference between two methods. Two third of the respondent categories assume that Agile is more testable than Waterfall.

4.4.3 CHANGEABILITY

Developer

The table below presents the analysis done on the developer's perception for quality factor changeability between Agile and Waterfall techniques.

$$H_0 - \mu_{Gap\_Developer\_Changeability} = 0$$

$$H_1 - \mu_{Gap\_Developer\_Changeability} \neq 0$$

Table 34: One Sample Statistics for Changeability – Developer

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Changeability	6.055	75	.000	.55000	.3690	.7310

Significant value in the Table 34 is less than 0.05 therefore we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no significant difference in Changeability between the software products developed using Agile and Waterfall techniques which means according to the developers 'there is a significant difference in Testability between the software products developed using Agile and Waterfall methods'

According to the values presented in the Table 34 the confidence interval of difference falls between .3690 and .7310. Since 0 does not fall within the interval and also both the limits are positive we can conclude that the differences are mostly positive. Since the differences are calculated by subtracting weight for Waterfall by the weight for Agile a positive difference implies Agile is better than Waterfall. Therefore we can conclude that from the perception of the developers software

products develop using Agile are more changeable than the products develop in Waterfall method.

Tester

The table below presents the analysis done on the tester's perception for quality factor changeability between Agile and Waterfall techniques.

$$H_0 - \mu_{Gap\_Tester\_Changeability} = 0$$

$$H_1 - \mu_{Gap\_Tester\_Changeability} \neq 0$$

Table 35: One Sample Statistics for Changeability – Tester

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Changeability	2.423	25	.023	.23462	.0352	.4340

Significant value in the Table 35 is less than 0.05 therefore we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no significant difference in Changeability between the software products developed using Agile and Waterfall techniques which means according to the developers 'there is a significant difference in Changeability between the software products developed using Agile and Waterfall methods'

As can be seen by the values presented in the Table 35 the confidence interval of difference falls between .0352 and .4340. Since 0 does not fall within the interval and also both the limits are positive we can conclude that the differences are mostly positive. Since the differences are calculated by subtracting weight for Waterfall by

the weight for Agile a positive difference implies Agile is better than Waterfall. Therefore we can conclude that from the perception of the developers software products develop using Agile are more changeable than the products develop in Waterfall method.

QA Lead

The table 36 below presents the analysis of gap between Agile and Waterfall on the perception of tester's for the quality factor changeability.

$$H_0 - \mu_{Gap\_QA\ Lead\_Changeability} = 0$$

$$H_1 - \mu_{Gap\_QA\ Lead\_Changeability} \neq 0$$

Table 36: One Sample Statistics for Changeability – QA Lead

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Changeability	.846	10	.417	.21818	-.3565	.7928

Significant value in the table 36 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Changeability between the software products developed using Agile and Waterfall techniques which means from the perspective of the QA Leads participated in this research 'there is no significant difference in Changeability between the software products developed using Agile and Waterfall methods'

As per the analysis, Developers and Testers believe that software products developed using Agile are more changeable than the products developed using Waterfall. In contrast the analysis done on the responses of QA leads reflected that there is no

significant different between the two methods in achieving the quality factor Changeability.

4.4.4 INSTALL-ABILITY

Developer

The table 37 below presents the analysis of gap between Agile and Waterfall on the perception of tester's for the quality factor changeability.

$$H_0 - \mu_{Gap\_Developer\_Installability} = 0$$

$$H_1 - \mu_{Gap\_Developer\_Instalability} \neq 0$$

Table 37: One Sample Statistics for Installability – Developers

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Installability	.877	75	.383	.07018	-.0893	.2296

Significant value in the table 37 given below, is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Installability between the software products developed using Agile and Waterfall techniques which means from the perspective of the developers participated in this research 'there is no significant difference in Installability between the software products developed using Agile and Waterfall methods'

Tester

The table below presents the analysis done on the tester's perception for quality factor Installability between Agile and Waterfall techniques.

$$H_0 - \mu_{Gap\_Tester\_Installability} = 0$$

$$H_1 - \mu_{Gap\_Tester\_Instalability} \neq 0$$

Table 38: One Sample Statistics for Installability – Testers

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Install-ability	-.891	25	.381	-.05128	-.1698	.0673

Significant value in the table 38 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Install-ability between the software products developed using Agile and Waterfall techniques which means from the perspective of the testers participated in this research 'there is no significant difference in Install-ability between the software products developed using Agile and Waterfall methods'

QA Lead

The table 39 below presents the analysis done on the QA Lead's perception for quality factor Install-ability between Agile and Waterfall techniques.

$$H_0 - \mu_{Gap\_QALead\_Installability} = 0$$

$$H_1 - \mu_{Gap\_QALead\_Instalability} \neq 0$$

Table 39: One Sample Statistics for Installability – Testers

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Installability	.886	10	.397	.12121	-.1837	.4262

Significant value in the table 39 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in Install-ability between the software products developed using Agile and Waterfall techniques which means from the perspective of the QA Leads participated in this research 'there is no significant difference in Install-ability between the software products developed using Agile and Waterfall methods'

As per the analysis done on the quality factor Install-ability none of the respondent categories believe that there is difference between the software products developed using Agile and Waterfall techniques. This demographic analysis further justifies the finding 'There is no significant difference between Agile and Waterfall techniques for the quality factor Install-ability' in the segment 4.3.4 page 96 of this document.

4.4.5 .PRODUCT QUALITY

Developer

$$H_0 - \mu_{Gap\_Developer\_ProductQuality} = 0$$

$$H_1 - \mu_{Gap\_Developer\_ProductQuality} \neq 0$$

Table 40: One Sample Statistics for Product Quality – Developer

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Product Quality	4.447	75	.000	.27270	.1505	.3949

Significant value in the Table 40 is less than 0.05 therefore we have to reject our null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is no significant difference in SW Product Quality between the software products developed using Agile and Waterfall techniques which means according to the perception of the developers 'there is a significant difference in Product Quality between the software products developed using Agile and Waterfall methods'

The confidence interval of difference (Table 40) falls between .1505 and .3949. Since 0 does not fall within the interval and also both the limits are positive we can conclude that the differences are mostly positive. As describe above positive difference implies Agile is better than Waterfall. Therefore we can conclude that Software product quality is high in SW products developed using Agile than the SW product developed using Waterfall.

Tester

$$H_0 - \mu_{Gap\_Tester\_ProductQuality} = 0$$

$$H_1 - \mu_{Gap\_Tester\_ProductQuality} \neq 0$$

Table 41: One Sample Statistics for Product Quality – Tester

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Product Quality	.790	25	.437	.04647	-.0747	.1677

Significant value in the table 41 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in product quality between the software products developed using Agile and Waterfall techniques which means from the perspective of the testers participated in this research 'there is no significant difference in product quality between the software products developed using Agile and Waterfall methods'

QA Leads

$$H_0 - \mu_{Gap\_QALead\_ProductQuality} = 0$$

$$H_1 - \mu_{Gap\_QALead\_ProductQuality} \neq 0$$

Table 42: One Sample Statistics for Product Quality – QA Lead

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Product Quality	.203	10	.843	.03182	-.3166	.3803

Significant value in the table 42 is greater than 0.05 therefore we do not reject the null hypotheses at 5% significance level. And conclude that there is no evidence to say that there is a difference in product quality between the software products developed

using Agile and Waterfall techniques which means from the perspective of the QA leads participated in this research ‘there is no significant difference in product quality between the software products developed using Agile and Waterfall methods’

According to the results presented, analysis done on developer’s responses reflected that there is a difference in achieving the SW product quality between the two methods. Further it stated that SW quality is high in Agile development compared to traditional Waterfall method. On the other hand results for testers and QA leads reflected that there is no significant difference in achieving the SW product quality between the two methods.

4.5 SUMMARY

The section begins with analyzing the validity of the data collected. There were 190 questionnaires distributed and 155 received in return. Out of the total received, only 113 were accepted to be considered at the analysis. Consequently, the respondent rate is 82% and the valid rate against the total received is 72.9 %.

The second part of the chapter paid attention on analyzing the collected data in order to test the derived hypothesis on the identified quality factors ‘Correctness‘ , ‘Testability’ , ‘Changeability’ , ‘Install ability’, ‘On time Delivery’ and ‘Complete within the budget’

In the process of analyzing gap between Agile and Waterfall techniques for the above mentioned factors, firstly, the questions related to each of the quality factors were studied separately. Then One sample T test was used against the calculated gap to check the validly of the hypothesis.

According to the analysis it has been identified that for the quality factors ‘Correctness’, ‘Install ability’, ‘Time’ and ‘Budget’ there is no significant difference between the two development methods. But for the factors ‘Testability’ and

‘Changeability’ there is a significant difference between the two methods. And for both the factors, the Agile method is better than Waterfall. The cumulative analysis for the product quality factors resulted that there is a difference in the quality of the software products developed using Agile and Waterfall techniques and further it states that Agile is ahead of Waterfall.

CHAPTER FIVE

5. DISCUSSION

5.1 INTRODUCTION

This chapter begins by summarizing and discussing the major findings of the research and it reiterates the findings of Chapter 4: Analysis. The chapter describes the importance of the findings and links it with other similar researches. Also it discusses any differences and similarities with the probable grounds for such differences. Further the section moves towards the limitations of the study and lastly it discusses the supplementary directions the research indicates.

5.2 REITERATION OF THE FINDINGS

The "one size fits all" approach to applying SDLC methodologies is no longer appropriate (Lindvall & Rus, 2000). Each SDLC methodology is only effective under specific conditions. Traditional SDLC methodologies are often regarded as the proper and disciplined approach to the analysis and design of software applications (Rothi & Yen, 1989). Examples include the Code and Fix, Waterfall, Staged and Phased development, Transformational, Spiral, and Iterative models. Agile techniques on the other hand are a compromise between no process and too much process. These new methods were developed to efficiently manage software projects subjected to short timelines and excessive uncertainty and change. (Lindvall & Rus, 2000). As described above many groups are trying to convince that Agile is the best methodology and there is no IT meeting that does not talk and debate endlessly about Waterfall vs. Agile development methodologies. Judgment run strong on the subject with many considering Agile just so right, while Waterfall is thought to be outdated. But, before deciding which method is more appropriate, it is essentially important to

make sure which method is quality wise sound. Hence overall idea of this thesis is to hit upon answers to the questions stated below.

- Is there a quality difference between the software products developed using Agile and Waterfall methods? And if there is a difference what is the healthier method?

In the process of identifying answers to this question the researcher focused on the quality factors mentioned below.

- Correctness
- Testability
- Changeability
- Install ability
- Complete within agreed time
- Complete within agreed budget

The findings for each of those quality factors are discussed further in this section.

Correctness

In the questionnaire questions 5 to 9 collects information that contributes to the quality factor 'Correctness' of a software product.

Q5 - The components we deliver almost meet user expectations

"Using agile modeling techniques and tools allows software developers to consider complex problems before addressing them in programming. Agile planning and development uses software modeling principles to let a developer design a software system that truly meets the customer's requirements. This will lead to develop a final product capable of catering the user's expectation" (Ambler, 2002, p.78). As described above according to Scott Ambler who offered a suite of principles and

practices for software modeling, it is easy to meet user expectations or customer requirements through Agility. According to Scott following factors make a contribution to the high achievement of user expectations in Agile development.

- Stakeholders actively participate in the agile planning and development
- Teamwork is established
- Appropriate artifact (such as UML diagrams) is used to create suitable models
- Several models are created in parallel
- Correctness of the agile software models is verified
- The verified models are implemented and the resulting interface is presented to the user
- Standards for agile requirement management are met

But in his study he has not considered traditional methods nor had compare Agile with Waterfall or any other traditional method. On the hand the analysis done, on the answers received for the question number 9 it has been identified by the researcher that most of the respondents agreed that same level of quality is achieved in meeting the user expectations in both the development methods. As mentioned earlier, due to the difficulty of getting information from the clients the research has collected data only from the development companies. Hence the analysis reflects only the organization's view point toward meeting their customer expectations but not direct information from the customers themselves. Thus some limitations could exist on the conclusions made on meeting user expectations.

Q6 - Our requirement specifications capture all the user requirements.

"Waterfall assumes that it is possible to have perfect understanding of the requirements from the start. But in software development, stakeholders often dosn't

know what they want and unable to articulate their requirements at once." (Szalvay, 2004)

"Agile methodologies embrace iterations. Small teams work together with stakeholders to define quick prototypes, proof of concepts, or other visual means to describe the problem to be solved. The team defines the requirements for the iteration, develops the code, and defines and runs integrated test scripts, and the users verify the results. Verification occurs much earlier in the development process than it would with waterfall, allowing stakeholders to fine-tune requirements while they're still relatively easy to change." (Szalvay, 2004)

As can be seen in the comparison done in the web article 'An Introduction to Agile Software Development' it seems that the argument is more towards Agile. But nowhere in the comparison has it been specifically stated that the requirement specification is sounder in Agile than in Waterfall. Thus the argument that since you can't come back to the previous phase in Waterfall, before moving in to next phase you thoroughly study the user's requirements and articulate a sound artifact. On the other hand in Agile since the process is iterative as mentioned above, it is not necessary to collect all the requirements at once and prepare a precise requirement specification (Anon, 2007). According to the results observed in this research majority of the respondents assume that there is no significant difference in capturing user requirements between the two methods. Of those who believe that there is a difference majority assume that Waterfall is better than Agile in capturing the user requirements in the requirement specification. This can be further justified by the statement in the article 'Testing Methodologies' published by Microsoft Corporation in January 2005.

"Working software is the priority rather than detailed documentation. Agile methodologies rely on face-to-face communication and collaboration, with people

working in pairs. Because of the extensive communication with customers and among team members, the project does not need a comprehensive requirements document" (Microsoft Corporation, 2005)

Q7 - Our system design cover the specifications 100%

According to the findings of this research most of the respondents assume that there is no significant difference between the two methods in capturing the specification in the design. But out of those who believe that there is a difference majority assume that Agile is better in incorporating the specifications in design. There were no written documents related to the quality attribute measured by this question.

Q8 - Our system implementation cover the system design 100%

As mentioned in Chapter 4 Analysis majority of the respondents falls on the fence assuming that there is no significant difference between the two methods in adapting the features in the design into the implementation.

Q9 - Our system implementation 100% free from faults

"By using Walkthroughs hidden implementation faults can be early detected, but a great effort is involved. Any development method can minimize the defects at the implementation if they incorporate walkthroughs within the development process". (Börcsök J, 2000/2001) According to Borcsok irrespective of the development method any product can achieve minimum fault % during the implementation if they adopt walkthroughs within the process. But he has neither stated a development method for which the walkthroughs are appropriate nor any comparison done against the process models. As per the analysis for this question most of the respondents have provided the same rating for both methods hence can conclude that most of the

respondents assume that there is no significant difference between the two methods when considering the amount of faults in the system implementation.

"Aside from refactoring and effective prototyping, agile methods have other advantages for a situation in which requirements are unstable. Reliance on test-first programming, a principle of XP, means early detection of most minor errors, more certain detection of defects at integration, and early thinking-through of tests for a Graphical User Interface. These features of the Agile techniques increases the correctness of the software products developed using the method." (Tomayko, 2002) The study done by Tomayko focuses only on the Agile Development and Specially XP. In his study he has not done any comparison between Agility and Traditional methods. Therefore based on his research it is difficult to decide on a better method to achieve the identified quality factor Correctness. But this section of the research aimed at identifying the most suitable method to achieve the quality factor Correctness. And the findings reflected that there is no significant difference in the two methods in achieving the above mentioned aspect.

Testability

After analysing the data received it has been identified by the researcher that there is a difference between the two development methodologies for the quality factor Testability and at the same time Agile is more testable than Waterfall.

The findings of the research can be further justified by the following interviews and research findings,

Since the Agile development consists of number of incremental iterations a product is tested many times before its ultimate release. But when a traditional method is considered like Waterfall, since development phases are chronological and cascaded

testing is done just once for the final product. (Phase-5). Therefore it is apparent that the testability of the software product is higher in Agile Development. (Talbi D et al, 2006)

"Agile testing is in close collaboration between the test writer and the developers to ensure test scripts can both be rapidly created and are robust and ensures flexibility and adaptability. It is an iterative process. In the typical classical approach test team is asked to test a project. Test engineers write the test cases. Testers need to wait for software until it is almost too late executing the test plans." According to Sugandhi following features of the Agile development facilitate in increasing the testability of its software products over the software products developed using traditional methods.

Pair programming – Each given task is handled by pair of programmers. One programmer writes the code while other one reviews code and highlights / comments the problems.

Test Driven Development – A 'bug' is anything that could bug a user. Testers don't make the final call. Testing along does not assure quality. Find ways to set goals rather than focus on mistakes. Developers write unit tests before coding. So it motivates coding, improves design (reducing coupling and increase cohesion), Support refactoring. Many open source test tools like xUnit have been developed to support this.

Refactoring – It is a way to improve the design of the existing code.ie changing a software system in such a way that it does not alter the external behaviour of the code, yet improves its internal structure make the simplest design that will work. Add complexity only when needed. Refactoring requires unit tests to ensure that design changes don't break the existing code.

Acceptance testing – User stories are short description of features that need to be coded. Acceptance tests verify the completion of user stories. Ideally they are written before coding. (Sugandhi et al., 2010)

Changeability

> "Unlike traditional methods Agile methods allow for specification changes as per end-user's requirements, spelling customer satisfaction. As already mentioned, this is not possible when the waterfall method is employed, since any changes to be made means the project has to be started all over again." (Scott, 2006)

> "The 'One Phase' and 'Rigid' development cycle makes it difficult to make last minute changes in requirements or design or the product. On the other hand agile methods, due to their iterative and adaptable nature, can incorporate changes and release a product in lesser time. Of course, agile models are not perfect either, but they are certainly more widely applicable than the waterfall model" (Pilgrim, 2010)

Roy Winston in his book "Managing the Development of Large Software Systems" has stated that in the Waterfall methods clients change their requirements after the design is finalized can kill the project. (Winston, 1970). But he has not compared the methods Waterfall and Agile the statement given by Winston only reflects that the Changeability is low in Waterfall development.

As has been quoted above almost all the surveys and researches has proven the fact that the Software products developed using Agile is changeable than the software products developed using Waterfall method. The findings of this research too has confirmed that there is a difference in changeability between the products developed using Agile and Waterfall techniques and at the same time it has verified that the changeability is high in Agile development compared to the Waterfall development.

Install-ability

Though there was lot of literature defining what install-ability is, no literature is found comparing this quality factor between Agile and Waterfall techniques. As described in Chapter 4: Analysis the findings of this research reflected that there is no significant difference for Install-ability between the software products developed using Agile and Waterfall methods.

Software Product Quality

Cumulative analysis of the above four factors that affect the Software Product quality suggested that there is a difference in Software Products developed using Agile and Waterfall methods and the product quality is higher in Agile Development in contrast to Waterfall development. Out of the four factors considered Changeability and Testability are more toward Agile hence these two factors might have strongly contributed to this judgment.

Above discussed findings of the research can be further justified by the "Agile Adaption Rate Survey result: February 2008" article presented by Jon Erickson the editor of the Dr Dobbs journal. As presented in the literature survey section 2.6 according to the findings of the survey only 9 % of the respondents assume that the product quality is lower in Agile development where as majority 48 % of the respondents assume that the product quality is much higher in Agile development.

Another article presented based on the research done on Waterfall Model Vs Agile by Gray Pilgrim stated that, "Through my own research into the working of both these models, I found the agile models to be more efficient and produce quality software products than the waterfall model, due to its adaptability to the real world. The 'One Phase' and 'Rigid' development cycle makes it difficult to make last minute changes in

requirements or design. While the agile methods, due to their iterative and adaptable nature, can incorporate changes and release a product in lesser time. Of course, agile models are not perfect either, but they are certainly more widely applicable than the waterfall model." (Pilgrim, 2010)

Complete within agreed time

As been discussed by Alberto in his article titled "Waterfall Vs Agile: Can they be Friends?" Time to market measures how fast a company can have a product out in the market from the moment they start developing. A fast time to market allows the company to have its product available long before its competitors. Agile is a sure bet to achieve very fast times to market as at the end of each iteration the application should be production ready. (Alberto 2010) But this assertion has not provided any comparison between Agile and Waterfall methods.

Article published by Toronto and Boulder on their study stated that "Larger software development teams, especially when geographically dispersed, often struggle to deliver their software on time. By adopting Agile practices, companies measured in this study were able to produce large-scale enterprise software in four to eleven months, compared to the six to thirteen months a typical organization required to deliver comparable software. Overall, Agile companies experience an average increase in speed of 37 percent. Customers who participated in the study saw an average increase of 50 percent in their time-to-market when compared to the industry average with the traditional methods. Here the authors have done a comparison between Agile and Traditional methodologies and has concluded that Agile is faster in delivering products compared to the traditional method. The reason for this has mentioned in Alberto's article Agile develops working software at the end of each iteration, whereas as mentioned in the literature survey section 2.6 (page 48, Agile

Impact Report) Waterfall develops the working product at the end of the entire development life cycle.

The finding of this research reflected that there is no significant difference between the two methods Agile and Waterfall in completing the project within the agreed upon time frame. As described above those two findings has considered only the 'time to market'. But this research focuses not on time to market but the capability of completing the project within the 'agreed time frame'. Perhaps this may be the reason for the difference between the literature and the conclusions of the researcher.

Complete within agreed Budget.

The result of the analysis of this study revealed that there is no significant difference between the two development models Agile and Waterfall for the identified project quality factor 'Complete within the agreed upon budget'.

Brad Egeland in his Agile Software Development Project Vs Standard Software Development Project white paper argued that the less re work and final product much closer to the end user requirement effects to the low project cost in Agile Development.

On the other hand Joe Ocampo states that "Agile produces higher value for the money but doesn't necessarily save you money in project cost." (Ocampo, 2007)

By analyzing both the studies it is apparent that no one has checked whether the Agile and Waterfall can complete the project within agreed upon budget frame but which method is cost effective. In contrast the target of this research is to discover whether there is a difference between Agile and Waterfall projects in completing projects within the agreed budget and if there is a difference which one is better. As mentioned

above the results reflected that there is no significant difference between Agile and Waterfall projects in completing within the agreed budget frame.

5.3 LIMITATIONS AND FURTHER RESEARCH

Limitations

1. As per the preliminary investigation it was impossible to retrieve customer oriented quality factors. Thus this research focused only on the developer oriented quality factors. These factors include 'Correctness', 'Testability', 'Changeability' & 'install ability' and two project quality factors Time & Budget.
2. As per the preliminary survey since it has been identified that it is impossible to collect data related to the process quality the research has not considered this factor in its study.
3. When considering the project quality the research has collected data only to check whether there is a difference in completing the project within the agreed upon time and budget using the development models Agile and Waterfall. But not the most effective method in relation to time to market and cost effectiveness.
4. The research has not differentiated its findings according to the size of the project due to the difficulty of the data collection within the given time frame.
5. The research was conducted in the Sri Lankan context and collected its data only from the Software Development Companies registered with the Sri Lanka Exports Association. So these results likely represent the experiences of IT professionals in Sri Lanka belongs to the above category, other Software Development companies around, neither the country nor other parts of the world has been measured.

Further Research

1. Further research can be conducted to analyse the other product quality factors discussed in the literature survey.
2. A research can be carried out to analyse the gap between Agile and Waterfall for the process quality factors
3. A study could be undertaken to capture the factors ‘Time to Market’ and ‘Cost Effectiveness’ between the two development methods.
4. Further research can also be conducted to identify the most suitable development method for large, medium and small sized projects.
5. The same research could also be conducted in the Sri Lankan context excluding the development companies registered with the SEA or in the context of another country.

CHAPTER SIX

6. CONCLUSIONS

6.1 INTRODUCTION

This chapter highlights the major findings of the research. The overall goal of this study is to identify the software development methodology that facilitates in producing high quality Software products in the Sri Lankan Context. The study was mainly around six developer oriented quality variables identified from the literature survey. Conclusions and recommendations of the gap analysis done between Waterfall and Agile techniques, against the identified quality factors, are described further in this chapter.

6.2 CONCLUSION

The section summarizes the findings for each identified quality factor namely Correctness, Testability, Changeability, Time and the Cost. Since the first two objectives of the research (To identify of the Software Quality Factors and To identify of the Traditional Software Development Models) is achieved through the literature survey this section does not spotlight on summarizing those factors again.

Correctness

There is no significant difference in Correctness between the software products developed using Agile and Waterfall methods.

Testability

There is a difference between the two methods for the quality factor Testability. The software products developed using Agile are more testable than the products developed using Waterfall.

Changeability

There is a significant difference between the two methods for the quality factor Changeability. Agile techniques are more changeable than Waterfall.

Install-ability

There is no significant difference in Install ability between the software products developed using Agile and Waterfall methods.

Complete within the agreed Time period

There is no significant difference in completing the project within the agreed upon time frame between the Agile and Waterfall techniques.

Complete within the agreed Budget

There is no difference in completing the project within the agreed upon budget between the development methods Agile and Waterfall.

Software Product Quality

There is a significant difference in the software product quality between the software products developed using Agile and Waterfall methods. Agile techniques are ahead of Waterfall method in providing quality software.

After analyzing the above findings we can conclude that there is a difference in software Product Quality including the quality factors Changeability and Testability between Agile and Waterfall methods. Whereas there is no significant difference between the two methods for the factors Correctness, Install ability, Time and Budget.

6.2 RECOMMENDATIONS

1. As the research indicates there is no significant difference between the two methods in completing the project within the agreed time frame. But as discussed in chapter 2, the Agile development facilitates to deliver fractions of the product within short development cycles. Therefore whenever it is required to have a working product within a short period irrespective of the total completion of the product it is recommended to use the Agile techniques rather than traditional Waterfall method.

2. Both the literature and the findings of this research proved that the changeability is high in Agile development. Therefore Agile techniques are recommended when developing complex software products where the requirements are not easily understandable and also in situations where the requirements are constantly changing.

3. The research also reflected that the testability is high in Agile development. Hence Agile method is recommended in the development of software products which need a thorough level of testing

4. The research has indicated that there is no significant difference for the quality factor 'Correctness' between the two development methods. Hence any software project needing only the above factor either method can be used

5. Further as there is no significant difference for the quality factor install-ability either method can be used in the projects where the install-ability is highly considered.

6. The research reflected that software products developed using Agile techniques were of higher quality than those developed using Waterfall techniques. Thus the Agile techniques are recommended in the process of developing Software products of a very high quality.

REFERENCES

1. Abrahamsson, P (2001). Rethinking the Concept of Commitment in Software Process Improvement. Scandinavian Journal of Information Systems, 13 (5). pp.69-98.

2. Alberto, G (2010). Waterfall Vs Agile: Can they be Friends. Available: http://agile.dzone.com/articles/combining-agile-waterfall Waterfall vs. Agile: Can they be Friends? Last accessed 2nd Feb 2011.

3. Ambler, SW (2002). Agile Modeling (AM) – Using Models to Carry Out the Development Process. Available: http://www.mymanagementguide.com/agile-modeling-am-using-models-to-carry-out-the-development-process/. Last accessed 5th Feb 2011.

4. Ambler, SW (Nov 2007). Is Agile Really that Successful? Dr. Dobb's' The World of Software Development.' 11 (2), 24-26

5. Anon, (2001) Agile Alliance Manifesto for Agile Software Development. Available: http://www.agilemanifesto.org/principles.html Last accessed 25th June 2010

6. Anon. (2005). Software Lifecycles. Available: http://www.wittmannclan.de/ptr/cs/rup_model.jpg. Last accessed 19th May 2010

7. Anon. (2007). An Introduction to Agile Software Development. Available: http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf. Last accessed 30th Sep 2011.

8. Anon. (2011). Glossary. Available: http://www.sei.cmu.edu/architecture/start/glossary/. Last accessed 29th Sep 2011

9. Anon (n.d). Sri Lanka Exports Association. Available: http://www.islandsoftware.org/ . Last accessed 10th June 2009.

10. Aoyoma & Mikio (1998). Web based Agile Software. Available: http://rockfish-cs.csunc.edu/COMP290-S02/Ayoma-98.pdf. Last accessed 3rd June 2010.
11. Balzer, R, Cheatham, T. E & Green, C (1983). Software Technology in the1990's: Using a New Paradigm. Computer. 16 (11), 39-45.
12. Baum, A. V (1983). Total Quality Control. 3rd Ed. New York: McGraw-Hill.
13. Beck, K & Andres, C (2004). Extreme Programming Explained: Embrace Change. 2nd ed. Boston: Addison-Wesley.
14. Benington, H. D (1983). Production of Large Computer Programs. Production of Large Computer Programs. 5 (4), 350-361.
15. Berander, P, Damm, L & Eriksson J (2005). Software Quality Attributes and Trade – offs. Ph.D Thesis, Blekinge Institute of Technology. Available: http://www.bth.se/tek/besq.nsf/(WebFiles)/5A52350A52726F51C12570A8004CB613/$FILE/Software_quality_attributes.pdf. Last accessed 27th Sep 2010.
16. Boehm, B & Turner, R (2003). Using Risk to Balance Agile and Plan-Driven Methods. Computer, 36 (6), 57-66.
17. Boehm, B (1988). Characteristics of Software quality. New York: North Holland Pub. Co American Elsevier.
18. Boehm, B (2006). A view of 20th and 21st century software engineering. ACM New York. Available: http://portal.acm.org/citation.cfm?id=1134288. Last accessed 10th May 2010.
19. Callen, T (2007). Emerging Markets Main Engine of Growth. [e-Book] : . Available: http://www.imf.org/external/pubs/ft/survey/so/2007/NUM1017A.htm. Last accessed 15 Aug 2010.
20. Cockburn, A & Highsmith, J (2001) Agile Software Development: The people factor, Available: http://www.Adaptivesd.com/Articles/IEEEArtical2Final.pdf. Last accessed 6th June 2010.

21. Crosby P.B (1979). Quality is free: The art of making quality certain, New York: New American Library.
22. Daniel.G (2004). Software Quality Assurance: From theory to implementation. 3rd ed. India: PERSON Education. 259
23. Deming W.E (1988). Out of Crisis: Quality, Productivity and Competitive position, United Kingdom: Cambridge University Press.
24. Dr. Daniel.G (2004). Software Quality Assurance: From theory to implementation. 3rd ed. India: PERSON Education. 259
25. Edwards, CD. (1968). The meaning of quality. Quality Progress. 6 (16) 80 -84.
26. Emanuel, R. Baker PD & Frank, J K (2000). SEI Capability Maturity Model. Available: http://www.qpmg.com/sei.htm. Last accessed 12th June 2009.
27. Erickson, J (2008). Agile Adaption Rate Survey result: February 2008. Available: http://www.ambysoft.com/surveys/agileFebruary2008.html. Last accessed 06th Jan 2011
28. Ferreira, C & Cohen, J (2008). Agile System Development and Stakeholder Satisfaction: A South African Empirical Study. Riding the wave of technology. Available: http://delivery.acm.org/10.1145/1460000/1456666/p48-ferreira.pdf?key1=1456666&key2=3955816821&coll=GUIDE&dl=GUIDE&CFID=104313137&CFTOKEN=66476760. Last accessed 6th June 2010.
29. Fitzgerald, B & Wynn, E ed. (2004). IT Innovation for Adaptability and Competiveness. Massachusetts: Kluwer Academic Publishers.
30. Gilb, T (1981). Evolutionary Development. ACM SIGSOFT Software Engineering Notes, 6 (2).2.
31. Gilb, T (1988). Principles of Software Engineering Management. Wokingham: Addison Wesley.

32. Gilb, T (2005). Competitive Engineering: A Hand book for System Engineering and Requirement Engineering. Butterworth: Heinemann.
33. Gilmore, H L. (1974 June). Product Conformance Cost. Quality Progress. 13 (2), 16-19.
34. Gould, P (1997). What is Agility? Manufacturing Engineer, 2 (5). 28-31.
35. Grady, R B (1992). Practical Software metrics for project management and process improvement. New Jersey: Prentice Hall.
36. Highsmith J (2001). History: The Agile Manifesto. Available: http://agilemanifesto.org/history.html. Last accessed 3rd June 2010.
37. Highsmith, J (1997). Messy, Exciting and Anxiety – Ridden: Adaptive Software Development. ACM Computing Surveys, 40(1). Available at: http://www.jimhighsmith.com /articles/ messy.htm. Last accessed 11th May 2010.
38. Highsmith, J (2000). Adaptive Software Development. Available: http://www.chc-3.com/cs511/fall2001/talks/team2_asd.ppt. Last accessed 11th May 2010.
39. Hower, R (1996). What is 'Software Quality Assurance?. Available: http://www.Software QA and Testing Resource Center - FAQ Part 1.htm. Last accessed 18th May 2010.
40. Hoyer RWH & Hoyer BBY (n.d). What is quality- Quality progress? The Global Voice of Quality. 34, 53-62.
41. IEEE Std. 610-12-1990. Available: http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf. Last Accessed 10th May 2010.
42. Ishikava K (1985). What is total quality control? The Japanese way, Englewood Cliffs: N.J. Prentice-Hall.

43. ISO/IEC (2001). Software Engineering – Product Quality, Part I: Quality Model, in ISO/IEC 9126-1:2001, International Organization of Standardization and International Electro technical Commission. USA.

44. James, T (2005). Stepping Back from Lean. IEE Manufacturing Engineer, 2 (3), 16-21.

45. Juran JM (1988). Quality Control Handbook, Boston: McGraw-Hill.

46. Juran, J M & Frank G (1988). Juran's Quality Control Handbook. Boston: McGraw-Hill.

47. Kokol, P. Zumer, V & Stiglic, B (1991). New Evaluation Framework for Assessing the Reliability of Engineering Software System Design Paradigms. In: Reliability and Robustness of Engineering Software II. 3rd ed. London: Southampton 173-184.

48. Krejcie, RV & Morgan, DW (1970). Determining sample size for research activities. Educational & Psychological Measurement, 30 (12). 607 - 610.

49. Kroll, P & Kruchten, P (2003). The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. 3rd ed. Massachusetts: Addison-Wesley.

50. Larman, C (1998). Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design. Meisterplanlos. 11 (C). 12-15.

51. Larman,C & Basili, VR (2003). Iterative and Incremental Development: A Brief History. Maryland: Thomas Publishing Co.

52. Leffler, K B. (1982). Ambiguous Changes in Product Quality. American Economic Review. 7 (3), 129.

53. Lewis, GF (1998). Unified process project profile. 4th ed. Massachusetts: Addison-Wesley.

54. Lindvall, M & Rus, I (2000). Process diversity in software development. IEEE Software, 17 (4). 14 - 18.

55. Lycett, M. Marcos, E & Storey, V (2007) Model Driven System Development: An Introduction European journal of Information Systems, 16. 346-348.
56. Madura, J (2007). Introduction to Business. USA: Thompson Higher Education.
57. Malouin, J L & Landry, M (1983). The Miracle of Universal Methods in Systems Design. Journal of Applied Systems Analysis, 10. 47-62.
58. McCall, JA. Richard, PK & Walters GF (1977). Factors in Software Quality. Nat'l Tech. Information Service. 1 (3). 2-3.
59. McCracken, D D & Jackson, M (1982). Life cycle concept considered harmful. ACM SIGSOFT Software Engineering Notes, 7 (2). 16-20. Available: http://portal.acm.org/citation.cfm?id=1005943. Last accessed on 2nd Aug 2010.
60. Microsoft Corporation. (2005). Testing Methodologies. Available: http://msdn.microsoft.com/en-us/library/ff649520.aspx. Last accessed 30 Sep 2011.
61. Morien, R (2005). Agile Management and the Toyota way for Software Management. In: 3 rd International Conference on Industrial Informatics. Perth WA Australia. 10 Aug 2005, IEEE: Perth.
62. Musa J I A & Okumoto, K (1990). Software Reliability. New York: Mc Graw-Hill.
63. Ocampo J. (2007). Agile Vs Traditional Development Cost Models. Available: http://lostechies.com/joeocampo/2007/09/20/agile-vs-traditional-development-cost-models-maybe/. Last accessed 30th Sep 2011
64. Pilgrim, G (2010). Waterfall Model Vs Agile. Available at: http://www.buzzle.com/articles/waterfall-model-vs-agile.html. Last accessed 04th Feb 2011.
65. Poppendieck, M & Poppendieck, T (2003). Lean Software Development: An Agile Toolkit. Boston: Addison-Wesley.

66. Pressman, P S (2010). Software Engineering: A Practitioners Approach. 7th ed. Boston: McGraw-Hill.

67. Ramand, G (2009). Software Quality Assurance. Quality Assurance and Software Testing, 11 (2). 32-34.

68. Ross, A & Francis, D (2003). Lean is not enough. IEE Manufacturing Engineer, 2 (4). 14-17.

69. Rothi, J & Yen, D (1989). System Analysis and Design in End User Developed Applications. Journal of Information Systems Education. Available at: http://www.gise.org/JISE/Vol1-5/SYSTEMAN.htm. Last accessed 2 Feb 2011.

70. Royce, W W. (1970), 'Managing the development of large software systems: concepts and techniques', Proc. IEEE WESTCON, Los Angeles, 1--9 .

71. Sanjay, AV (2005). Overview of Agile Management & Development. Available: http://www.projectperfect.com.au/downloads/Info/info_agile _programming.pdf Last accessed 4th June 2010.

72. Saparamadu D B (n.d.). Overview of Sri Lankan IT Industry. Available at: http://www.hsenid.com/download/EH_Overview_Of_the_Sri_Lankan_ITDetail.pdf. Last accessed 12 May 2010.

73. Satalkar, B(2011). Waterfall Model Advantages. Available: http://www.buzzle.com/articles/waterfall-model-advantages.html. Last accessed 28th Sep 2011.

74. Schauble, W (2007). Advancing e-government. In: German EU Council Presidency. International e-Government Conference. Berlin Germany. 01 -02 Mar 2007, Federal Ministry of the Interior: Berlin.

75. Schwaber, K (2004). Agile Project Management with Scrum. Washington: Microsoft Press.

76. Schwallbe, K (2004). Information Technology Project management. 3rd ed. USA: Thompson learning Inc.

77. Scott, A (2001). When and when aren't you agile modeling? Available at: http://www.agilemodeling.com/essays/whenAreYouAgileModeling.html. Last accessed 6th June 2010.

78. Smith, P (28 June 2011). Waterfall SDLC Methodology. Available: http://skysigal.xact-solutions.com/Resources/SoftwareDevLifeCycle/WaterfallMethodSDLC/tabid/600/Default.aspx. Last accessed 29th Sep 2011.

79. Stapleton J (2003). DSDM Consortium. 2nd ed. London: Addison- Wesley. 176.

80. Sugnadhi, AD. Anand, N. Manisha, V & Garje JV (2008). Improving Software Quality with Agile testing. International Journal of Computer Applications. 1 (3), 22.

81. Szalvay, V (2004). An Introduction to Agile Software Development. Available at: http://docs.google.com/viewer?a=v&q=cache:5HFgofvhZVIJ:www.danube.com/docs/Intro_to_Agile.pdf+An+Introduction+to+Agile+Software+Development&hl=en&pid=bl&srcid=ADGEESivrgmS.html. Last accessed on 10 June 2010.

82. Toronto & Boulder (2008). The Agile Impact Report: Proven Performance Metrics from the Agile Enterprise. Available at: http://www.pr-inside.com/new-study-shows-that-agile-teams-r737178.htm. Last accessed 07th Feb 2011.

83. Tyrell, S (2000). The Many Dimensions of the Software Process. Cross Roads: The ACM Student Magazine 6(4), 22-26. Available at: http://portal.acm.org/citation.cfm?id=333435. Last accessed 07th Feb 2011.

84. Williams, L & Cockburn, A (2003). Agile Software Development: It's about feedback and change. Computer. 36 (6). 39-43.

85. Witmann, P (2005). WittmannClan.com: Software Life cycles. Available at: http://www.wittmannclan.de/ptr/cs/slcycles.html. Last accessed 25 Aug 2010.
86. World Economic Outlook Database. Available: http://en.wikipedia.org/wiki/File:Gdp_accumulated_change.png. Last Accessed 12 May 2010.
87. Zhu, H (2007). Quality-Attribute Auditing: The What, Why, and How. Available at: http://msdn.microsoft.com/en- us/library/Bb508961.0705 auditing _01l (en-us, MSDN.10).gif. Last accessed 12th June 2009.

APPENDIXES

Company Name: ..…….

Name of the Contact Person: ……

Designation: ……..

No of years in current position: ……

	1 Waterfall	**2 Spiral**	**3 RUP**	**4 Agile**	**5 Other**
1.What are the SW development process models use by your company					
2. Do you have any completed projects in each model					
3. Number of projects completed in each process model					
4. Number of on going projects in each process model					

5. If your answer is 5(Other) for question 1 please name the process model(s) use by your company in its SW development process.

   i. **.....** iii. …….

   ii. …….. iv. …….

6. Is it possible to collect required information from your company for the research 'Software Quality Assurance in Agile Development'?

   Yes No

Comment ……...

……...

7. Is it possible to contact your clients to get information on product quality?

   Yes ☐ No

Comment ……...

……...

	1 **Scrum**	**2** **ASD**	**3** **XP**	**4** **Crystal**	**5** **Other**
1. What agile methodologies use in your organization in its software development process.					

2. If your answer is 5(Other) for question 1 please name the methods.

i. .....
ii. .....
iii. .....
iv. .....

Please specify the number of employees for each category

Respondent Information

1. Company Name: ..……
2. Name of the Contact Person: ……
3. Designation: ……

Number of Employees

Designation	Count
PM	
QA Lead	
Developer	
Tester	

Questionnaire
Software Quality Assurance in Agile Development

Please corporate with us to complete this Survey. We do not require your identity but your suggestions and ideas are highly appreciated.

Please mark with ✓ on the relevant boxes/places.

Thank you very much for your kind corporation

---

1. Company: ☐ Virtusa ☐ TeamWorks ☐ DMS ☐ E - College

2. Have you completed at least one project using Waterfall method ☐ Yes ☐ No

3. Have you completed at least one project using Agile ☐ Yes ☐ No

4. Designation: ☐ Developer ☐ QA Team Lead ☐ Tester

Please use the below weightings to answer the questions 5 to 20

Strongly Disagree	1
Disagree	2
Neutral	3
Agree	4
Strongly Agree	5

If any question given below is not applicable to you please specify it with a ✓ in the "**NA"** column

		Agile					**Water fall**					**NA**
		1	2	3	4	5	1	2	3	4	5	
5	The components we deliver almost meet user expectations.											
6	Our requirement specifications capture all the user requirements.											
7	Our system design cover the specifications 100%											
8	Our system implementation cover the system design 100%											
9	Our system implementation 100% free from faults											
10	We accomplish complete execution of test scripts											
11	We always adhere to the Coding standards in implementing our systems.											

		Agile					**Waterfall**					**N**
		1	2	3	4	5	1	2	3	4	5	**A**
12	We do not employee many complex structures in our codes											
13	Modifications can be done to our products without much difficulty											
14	Our systems do not need much effort to accommodate minor specification changes											
15	Our systems maintain law interaction between modules											
16	When the changes are done to one module our systems have very low side effects to other modules											
17	Integration of a new component to the system does not create much functional issues											
18	Our systems do not challenge during the installation in the agreed environment.											
19	We have to incorporate minor modifications when installing our systems in the agreed environment											
20	Hardware configuration is always compatible with software we developed when deploying our systems											