

**AN EMPIRICAL STUDY ON THE ROLE OF REQUIREMENT
ENGINEERING IN AGILE METHOD AND ITS IMPACT ON QUALITY**

ANZIRA RAHMAN

**A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ARTS**

**GRADUATE PROGRAM IN INFORMATION SYSTEMS AND
TECHNOLOGY**

**YORK UNIVERSITY
TORONTO, ONTARIO
APRIL 2015**

© Anzira Rahman, 2015

ABSTRACT

Agile Methods are characterized as flexible and easily adaptable. The need to keep up with multiple high-priority projects and shorter time-to-market demands could explain their increasing popularity. It also raises concerns of whether or not use of these methods jeopardizes quality. Since Agile methods allow for changes throughout the process, they also create probabilities to impact software quality at any time. This thesis examines the process of requirement engineering as performed with Agile method in terms of its similarities and differences to requirement engineering as performed with the more traditional Waterfall method. It compares both approaches from a software quality perspective using a case study of 16 software projects. The main contribution of this work is to bring empirical evidence from real life cases that illustrate how Agile methods significantly impacts software quality, including the potential for a larger number of defects due to poor non-functional requirements elicitation.

ACKNOWLEDGMENTS

It has been an enlightening journey from the inception till the completion of this master thesis. During all this time of ups and downs, I was encouraged and given a lot of support by a lot of people, which I would like to thank with all my heart and sincere feelings.

I would like to thank my supervisor Professor Luiz Maricio Cysnerios for giving me this opportunity and having faith in me. I am compelled to say that without his encouragement and enthusiastic support, this thesis would not have been possible. I can never thank him enough for his academic support and help.

TABLE OF CONTENT

ABSTRACT.....	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENT	iv
LIST OF TABLES.....	vii
LIST OF FIGURES	viii
CHAPTER 1 INTRODUCTION.....	1
1.1 Requirement Engineering: Its Definition and Value in Corporate Settings.....	1
1.2 Software Quality Assurance: Its Definition and Value in Corporate Settings	2
1.3 Various Software Methods.....	3
1.4 Purpose of the Research	4
1.5 Research interest and motivation	5
1.6 Thesis Contribution	5
1.7 Thesis Organization	6
CHAPTER 2 REVIEW OF RELEVANT LITERATURE.....	7
2.1 Software Development Method.....	7
2.1.1 Agile Development Method	8
2.1.2 Waterfall Development Method	10
2.1.3 Comparison of Agile and Waterfall Development Methods	11
2.2 Requirement Engineering (RE)	12
2.2.1 Types of RE	13
2.2.2 RE in Agile	13
2.2.3 RE in Waterfall.....	14
2.2.4 Requirement Engineering in Agile Versus Waterfall Methods.....	15
2.3 Software Quality Assurance	17
2.3.1 Software Quality Assurance in Agile	18
2.3.2 Software Quality Assurance in Waterfall	20
2.3.3 Comparison of Software quality between Agile and Waterfall Method.....	21
2.4 Summary	23
CHAPTER 3 METHODOLOGY.....	24

3.1	Rationale for Selecting Case Study Research	24
3.2	Types of Case Study Research	25
3.3	Qualitative and Quantitative Research	27
3.4	Summary	28
CHAPTER 4 RESEARCH METHODS AND DATA COLLECTION		29
4.1	Hypotheses of the Research Project	29
4.2	Data Collection Strategy	30
4.3	The Organization	30
4.4	Formation of the Agile and Waterfall Group	31
4.5	The Source of Data	32
4.6	Criteria for Case Selection:	34
4.6.1	Project Size:	34
4.6.2	User Stories:	35
4.6.3	Work Items:	36
4.6.4	Team Size:.....	37
4.6.5	Duration:.....	38
4.7	Description of Agile and Waterfall Projects	40
4.8	Outcome Variables:	43
4.8.1	Defects:.....	43
4.8.2	Number defects:.....	44
4.9	Threats to Validity	45
4.9.1	Threats to Internal Validity.....	45
4.9.2	Threats to External Validity	45
4.10	Statistical Approach	46
4.11	Summary	47
CHAPTER 5 RESULTS AND DISCUSSIONS		48
5.1	Analysis of User Stories and Defects in Agile Group	49
5.2	Analysis of Work Items and Defects in Waterfall projects:.....	50
5.3	Comparison between Agile and Waterfall Projects	52
5.3.1	Hypothesis 1:.....	52
5.3.2	Hypothesis 2:.....	54
5.4	Strengths	57

5.5	Limitations.....	58
5.6	Summary	59
CHAPTER 6 CONCLUSION AND FUTURE WORK.....		60
REFERENCE.....		62

LIST OF TABLES

Table 1: Types of Case Studies.....	26
Table 2: Relationship between Year of Project Completion and Group Assignment.....	31
Table 3: Sample report of defects in a project	34
Table 4: Sample format of a user story and acceptance criteria.....	35
Table 5: Description of Agile projects.....	41
Table 6: Description of Waterfall projects.....	43
Table 7: Group Statistics of Agile and Waterfall.....	52

LIST OF FIGURES

Figure 1: A comparison of Agile and Waterfall life cycles [32].....	12
Figure 2: The relationship between Agile development methods and quality assurance techniques. [32].	19
Figure 3: Stages and SQA activities in Waterfall [3].....	21
Figure 4: The Empirical Cycle of a Case Study [53].....	25
Figure 5: Generic query used to execute the report from TFS.....	33
Figure 6: Size of selected projects using an Agile method	37
Figure 7: Size of selected projects using an Agile method	37
Figure 8: Team size, project duration and project size for each Agile project selected for analysis	39
Figure 9: Team size, project duration and project size for each Waterfall project selected for analysis	39
Figure 10: Agile Projects: Proportion of User Stories and Defects	49
Figure 11: Agile Projects: Defects with proportion of NFR and FR related defects	50
Figure 12: Waterfall projects: Proportion of work items and defects	51
Figure 13: Waterfall projects: Defects proportion of NFR and FR related defects	51
Figure 14: Mean of Number of defects for Agile and Waterfall.....	53
Figure 15: Mean of NFR-related Number of defects for Agile and Waterfall.....	55

CHAPTER 1 INTRODUCTION

In information-intensive societies, such as North America, technological solutions have become a popular trend. Such societies have a high expectation of generating economic growth, innovation, and creating new ways to distribute knowledge. As a result, more efficient ways of software development are being applied and evaluated in team settings in companies across the globe. Although the efficiency and faster turnaround are expected, software quality cannot be compromised. This chapter provides the background and rationale for this assessment of the impact of different software development methods on the quality of its output.

1.1 Requirement Engineering: Its Definition and Value in Corporate Settings

Requirement Engineering (RE) is a process of formulating, documenting and maintaining software requirements. In this process, the software need is identified; without this process, achieving great success in delivering functional software may be impossible. Dealing with requirements used to be the first order of business and the process took place before design, cost estimation, planning or programming to satisfy quality levels [5]. However, in recent times, requirements are analyzed throughout the software development life cycle. Though requirement engineering is time consuming, it is the only way to maximize the chances for the delivered software to meet the specifications [5]. Thus, RE plays a key role in software development.

In RE, functional requirements (FR) and non-functional requirements (NFR) are both crucial to the success of the project. While FR refers to features the software should provide,

NFR refers to constraints imposed on these features that could be local to one specific feature or applied to the overall operations of the system. An example of an FR would be “the system must send an email when an order is placed”. A related NFR would be “emails must be sent with a latency of no greater than 1 hour of that activity”. Though NFR elaborates a performance characteristic of the system, it is often ignored in software design [13]. By ignoring key requirements, the success of software cannot be accomplished.

1.2 Software Quality Assurance: Its Definition and Value in Corporate Settings

Software Quality Assurance (SQA) is an integral part of any software development method. It refers to a series of procedures that stipulate the desired level of quality in software, especially by means of attention to its feature functionalities and overall operations. As Non-functional Requirements (NFRs) are often overlooked in RE, their absence tends to result in failures and very expensive and difficult to correct [13]. Minimizing mistakes or defects and building software to meet the customer’s requirement(s) have always been highly valued and can lead to significant savings in software life-cycle costs [43] [7]. Though, the definition of SQA has changed over time, meeting the needs of the customers and providing product satisfaction has always been a primary consideration in SQA [28]. Software reliability, stability, and usability result in greater customer satisfaction.

1.3 Various Software Methods

Since the late 1960s, many different software development methods have been introduced and used by the software engineering community [32][44]. These methods have been improved and refined over time [32]. Some of them have been in existence longer than others and reached a more stable level and are referred to as “traditional” software development methods.

The Waterfall method is one of the many traditional methods and involves a software lifecycle that is divided into linear stages [32]. In contrast, the Agile software development method is more modern and comprised of a group of activities that encourages dynamic planning, development, and frequent delivery of artefacts to meet client satisfaction [4] [25] [18].

Since Agile software development methods focus on minimal documentation, Paetsch et al. [31] have suggested that Agile and Waterfall software method seem contradictory. In fact, Traditional and Agile RE are similar with respect to the performance of primary activities, such as elicitations, management, and validations, but differ in terms of when these activities are executed. More specifically, traditional RE is dependent on documentation for knowledge sharing, whereas Agile methods are less document-centric and prioritize face-to-face collaboration. In Agile process, teamwork is conducted within a short timeframe to gather requirements, develop, test, and deliver [25]. Overall, abilities such as accommodating changes, enhancing customer relationships, greater return on investment, and shorter development periods are identified as key factors for the increasing demand for agile practices [45] [31]. Thus, one well-recognized benefit of Agile RE has been the ability to create more satisfactory relationships with customers due to its quick delivery of high business value features and easier accommodation for changes in requirements [45]. Though, in Agile method, validations happen

throughout the process to ensure quality [18] [32], the easier accommodation of changes in Agile RE is typically completed without sufficient documentation, which may lead to challenges in maintaining software quality.

1.4 Purpose of the Research

The aim of this research project was to conduct an empirical test of whether Agile methods benefits over Traditional methods in terms of the number of defects, including functional and non-functional related defects. According to anecdotal reports, Agile methods, such as Scrum, are unable to produce high-quality products due to its volatile nature, however, the Agile process has been described by the Standish group as the “universal remedy for software development project failure [46]. To date, there has not been sufficient study regarding the actual practices of software professionals for software activities using Agile methods [24]. This project used case study methods to examine data from real-life scenarios. The following research questions were asked:

1. Is there a significant difference in the number of defects/issues produced in Agile and Waterfall methods?
2. Is there a significant difference in the number of NFR-related or FR-related defects/issues produced in Agile and Waterfall methods?

1.5 Research interest and motivation

Apart from the trend with insufficient research work, being working in a team who primarily offers technological solutions to clients and experiencing the shift of methods from Waterfall to Agile raises interests to further dive into the topic of whether it is the better solution for future of software methods or not.

1.6 Thesis Contribution

This study makes several contributions. First, although the critical nature of software failure and success has been investigated for years, this study makes further contributions by providing a detailed analysis of the benefits of Agile vs. Waterfall methods for a small to mid-size software team of 30 members. Second, despite Agile method's popularity in the real world, its benefits and risks have not been empirically tested using real life cases that emerged in real life situations. Until now most of the works addressing Agile methods rely on judgment calls made by developers using the method and their experience only. In fact, some of the aspects are still in an exploratory stage. In order to overcome this gap, this project is one of the first to offer empirical evidence of the impact of Agile methods on software quality. Lastly, since most organizations are looking to bring in efficient systems/procedures regarding RE and software, these findings may be useful to companies struggling to better understand the complex comparisons between Agile and Waterfall methods while deciding whether to adopt Agile methods.

1.7 Thesis Organization

This thesis is organized into six chapters. In Chapter 1, an introduction to the issue of Software Quality on Agile vs. Waterfall software method and the purpose of the research study is presented. Chapter 2 provides a review of the literature regarding Agile methods, requirement engineering in Agile and Waterfall, and software quality, including a summary of all relevant terminology. Chapter 3 elaborates on the quantitative methodology used in this study, its rationale, and the method employed in the process of data collection. The hypotheses and the data are analyzed and presented in Chapter 4. In Chapter 5, the study findings, the findings of similar research, strengths and limitations are discussed. And finally, the thesis is summarized, and possible future work is presented in Chapter 6.

CHAPTER 2 REVIEW OF RELEVANT LITERATURE

This chapter provides background information about the two software development methods (Agile and Waterfall) under investigation in this study. First, the software development method is defined, and then, the characteristics, and benefits of each are described. Next, the definitions of requirement engineering and software quality assurance are introduced, and their application in both methods is discussed. Throughout the chapter, a comparison of the Agile and Waterfall method reveals their similarities and differences in each area (e.g., benefits, requirement engineering, and procedures for quality assurance).

2.1 Software Development Method

Over the years, many systematic, disciplined and quantifiable methods of software development have been introduced. Each of these methods involves multiple phases and a variety of activities. For example, design, re-factor, re-use, re-engineering, and maintenance are some of the common activities employed to accomplish software solutions [31]. The goal of these activities is to develop software that meets the customers' needs.

In last 50 years, several methods have been introduced to enhance the success in software development. Some of the common methods include Waterfall, Prototyping, Iterative and Incremental Development, Spiral, Rapid Application Development, Extreme Programming, and Agile. The selection of a software development method is significant in any software project because strengths and weaknesses of the method have an impact on the customer [20]. Some researchers have proposed that the choice of a particular method is related to a number of

situational factors, including organizational, project-related and team-related factors [36], and therefore, a single method is not suited for all software development projects. The decision of using a specific method can be based on the literature and the standards of consistency in the industry [51]. In the next section, the characteristics and benefits of Agile and Waterfall development method are described.

2.1.1 Agile Development Method

The Agile Development method emerged as the result of a gathering of seventeen representatives from the software development industry in Snowbird, Utah in 2001 [2]. Their aim was to promote innovative approaches to software development that would enable organizations to react quickly and adapt to changing requirements and technologies. In their Agile manifesto [2], they identified the following four priorities: (a) individuals and interactions over processes and tools, (b) working software over comprehensive documentation, (c) customer collaboration over contract negotiation, and (d) responding to change rather than following a plan. The manifesto is based on twelve principles focusing on gaining customer satisfaction by frequently delivering working software and by being in a self-organizing team. There are many types of Agile methods, such as extreme programming, scrum, feature-driven development, dynamic system development method, adaptive software development, and crystal and lean software development [4] [9] [50] [35]. Common to all methods is the division of clients' needs into multiple release cycles that are presented in smaller portions according to their business value [52] [20]. These methods include some of the most recognizable quality factors such as: cost-effectiveness, efficiency, extendibility, maintainability, portability, reusability, and

robustness [28]. Overall, Agile encourages software development teams to remain as flexible as possible while working towards their common goal of producing a successful result.

Today's dynamic and fast-paced business environment is increasingly choosing Agile methods over other methods due to its benefits [9] [20] [25] [31] [36] [45] [51]. Due to the ability to handle volatile requirements throughout the software development life cycle and the ability to deliver in a shorter timeframe, agile methods have gained acceptance [32]. The major benefits of Agile methods are: 1) reduced risk and 2) increased revenue.

In Agile methods, risk of failure is reduced by the iterative approach with frequent continuous planning and feedback. While the team needs to remain focused on delivering an agreed subset of the product's features during each iteration, there are opportunities to constantly refine and reprioritize the overall product backlog. New or changed backlog items can be planned for the next iteration and provide the opportunity to introduce changes within a few weeks' time. Regardless of the size of a project, customer response is quick and frequent. Though according to the Standish Group, the Agile method is considered one of the factors for the success of small projects as it embodies the philosophy of the small projects [46], larger projects can also be broken into smaller projects and delivered in an iterative process.

As the risk is reduced, the revenue increases as well by having clear understanding of priorities. It supports reduced overhead cost in creating, re-doing, and maintaining requirements [25] [31] and helps gain more revenue. The active involvement of the user or business representatives during the software development process creates a positive working relationship and a more accurate understanding of project goals. For example, Liu et al.'s [27] survey of 377 participants from Chinese software companies and researchers revealed that the three reasons for

major failures were an unclear understanding of the systems, not understanding constant change, and not having enough interactions with customers. Apart from delivering portions of products more frequently, the Agile method supported working features as the most important measure of progress throughout the project life cycle to bring more revenue in a shorter timeframe [7]. For example, re-prioritizing, adding and modifying existing requirements are common in Agile because they are viewed as contributing to cost minimization [7] [25]. Agile brings efficiency in the process to make the most of the resources and maximize profit.

2.1.2 Waterfall Development Method

Over the years, many software development methods have been refined and referred to as “traditional” methods. One of oldest of these traditional methods is Waterfall, which was first described by Winston Royce in 1970 [40] and is still widely practiced in both small and large projects [3]. The Waterfall development method consists of sequential phases, including requirement analysis, software design, implementation, testing, integration, deployment, and maintenance. Typically, in this model, activities are seen as more steadily flowing downward, and more emphasis is given to planning, time schedules, target dates, budgets, and implementation of an entire system at once. This model prioritizes that the team has an understanding of the complete requirements of a project before beginning because plans are not re-evaluated during the development process.

Historically, there have been many successful projects completed under Waterfall. It has been used widely in both small and large projects [3], and many benefits have been reported.

Some of the key benefits of Waterfall are: promote efficiency with stable project definition and easier management of due to the rigidity of the each phase. Waterfall method is known for its more concrete and complete requirements, and these characteristics make this approach more stable. As mentioned earlier, the first of five stages of this method is requirement analysis. In this method, it is believed that spending more time early in the cycle can lead to greater success at later stages. Also, management of Waterfall method is easier due to the linear approach.

2.1.3 Comparison of Agile and Waterfall Development Methods

Agile and Waterfall methods differ in terms of the types of activities and when they are used within the development process. According to Huo et al. [32], the Agile approach turns the traditional software process “sideways”. Whereas traditional plan-driven development methods such as Waterfall use distinct and linear stages such as requirements analysis, system design, implementation, testing, and maintenance, the Agile methods involve overlapping processes [3] [32] .

Figure 1[32] shows that the activities in Waterfall, such as ‘Requirement analysis and definition’, ‘System and software design’, ‘Implementation and unit testing’, ‘Integration and system testing’ and ‘Operation and maintenance’, take place one after another in sequential order. However, in Agile methods, after the ‘creation of user stories’ and ‘release planning’, activities such as ‘iteration planning’, ‘create unit test’, ‘develop code’, ‘continuous integration’, ‘acceptance test’ take place simultaneously.

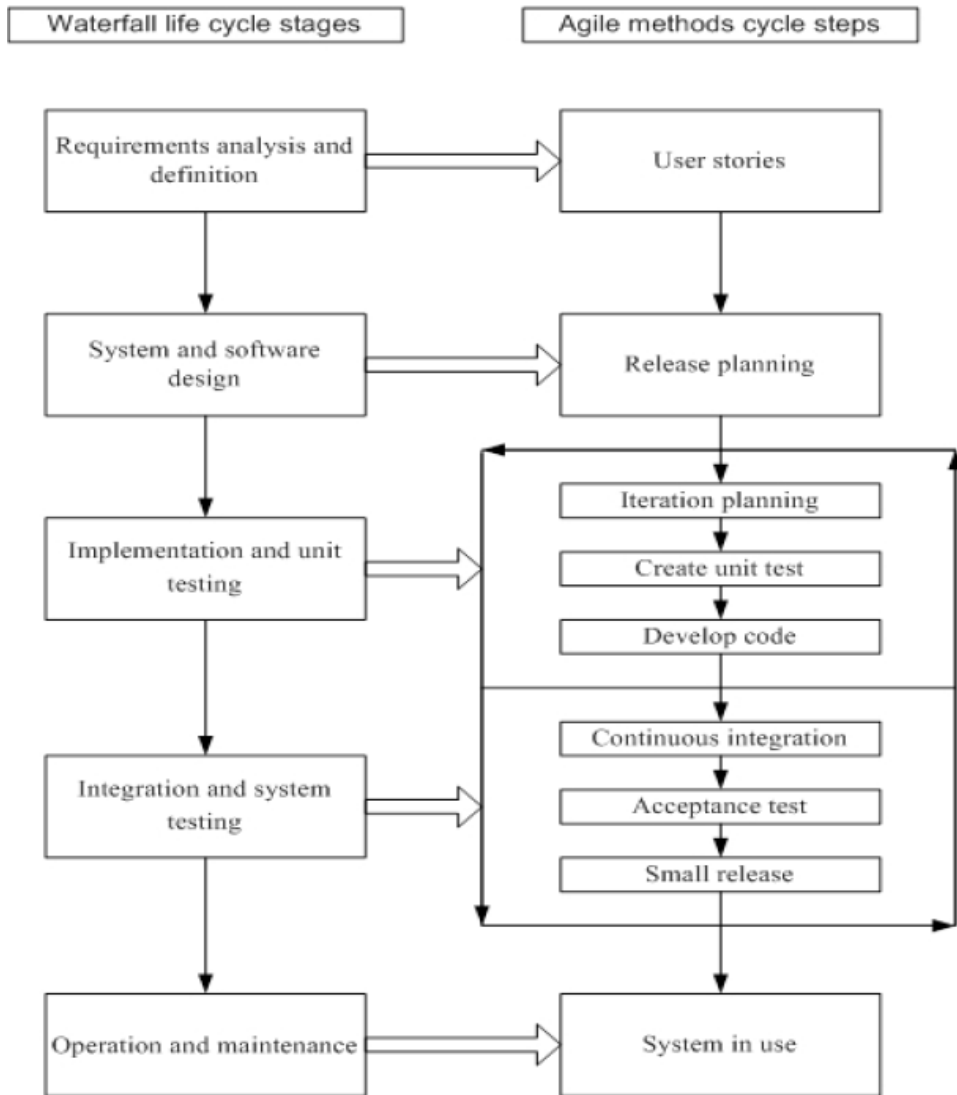


Figure 1: A comparison of Agile and Waterfall life cycles [32]

2.2 Requirement Engineering (RE)

Requirement engineering is a knowledge intensive activity that involves eliciting, analyzing, documenting, validating, and managing software requirements [4] [18] [25] [31].

According to Thayer [49], Requirements Engineering (RE) provides the appropriate mechanism to understand what the customer wants [18] and helps to analyze their needs, so that the

feasibility of the software can be determined and solutions can be negotiated. According to International Requirements Engineering Board, RE is a disciplined approach that is performed by documenting and managing stakeholders' needs in systematic ways [46].

2.2.1 Types of RE

Requirements can be classified into two categories: Non-functional requirements (NFR) and Functional Requirements (FR). NFR are also known as quality requirements and refer to criteria that can be used to qualify the operation of a system and to define how software should behave at any given time. Some of the NFR includes reliability, security, accuracy, safety, performance, look and feel requirements, as well as organizational, cultural and political requirements [13]. On the other hand, FR refers to specific behaviors that are involved with the function of the software to define what features and functions it should provide. For example, FR may include calculation, data manipulation, and any specific business rule regarding processes. In the next section, its briefly discussed how RE is performed in Agile and Waterfall methods are described.

2.2.2 RE in Agile

In Agile, RE is achieved through continuous collaboration while requirement gathering, developing and testing may happen at the same time. Many software development organizations now deal with requirements that tend to evolve quickly and become obsolete even before project

completion [5]. Hence, the need for continuous RE is considered necessary for successful delivery in these cases. Gallardo-Valencia and Sim [18] mentioned that recent studies have shown that requirement engineers prefer short requirements. In a field study of a company with 26 members practicing Agile, the requirements were captured incrementally by the teams and communicated verbally rather than using a comprehensive document to summarize requirement knowledge [18]. In Agile, the business requirements are elicited and documented in the form of user stories, which are from portrays user's perspective [46]. These user stories are used as a primary unit of work and continue to grow during the lifecycle of the project. Cao and Ramesh [25] performed a qualitative study of RE practices in 16 organizations that practice Agile and reported that its key practices included: face-to-face communication, iterative requirement engineering, extreme prioritization, constant planning, test-driven development, reviews, and tests. Though Agile developers insist on directly involving users and delivering “product versions” in short increments, the need for requirements is very critical [35]. Validating these requirements also continue during RE in Agile.

2.2.3 RE in Waterfall

In Waterfall method, RE takes place in the first phase of the development process and makes use of the following five activities: elicitation, analysis and negotiation, documentation, validation, and management [31]. In requirement elicitation, consulting with stakeholders discovers the process requirements and in requirement analysis, the elicited requirement's consistency and completeness are ensured [7]. Documentation, requirement validation, and management are also done systematically.

2.2.4 Requirement Engineering in Agile Versus Waterfall Methods

Though the purpose of RE is the same in all software methods, how RE is performed in Agile and Waterfall methods is opposing in nature [4] [31]. In this section, RE in Agile and Waterfall are discussed in terms of their differences with respect to customer involvement, non-functional requirements, as well as the use of an iterative approach, prioritization, and documentation.

With respect to customer involvement, RE in both Agile and Waterfall methods place importance upon stakeholder involvement, face-to-face collaboration, and delivering high-quality products [18] [31]. However, while customer collaboration and communication are crucial for achieving a good quality product [25] [18] [7], Agile and Waterfall methods differ in terms of when customers are consulted. In Waterfall, customers are involved during the initial phases of requirement gathering and analysis, whereas, in Agile, customers are involved throughout the entire process [31].

Agile and Waterfall methods are also different with respect to their use of Non-Functional Requirements. In Agile approaches, NFRs are not properly identified, modelled or linked during the early requirement analysis phase [14] [15]. Due to the nature of evolving requirements, NFR such as resources, maintainability, portability, safety or performance are often overlooked in Agile [31]. In addition, NFRs are not often seen as crucial or highly critical by teams using Agile methods [34] [13]. In contrast, there are some approaches to deal with NFR that are more likely to be used in waterfall where models and design are carefully developed; then NFR might be better handled in Waterfall.

The use of an iterative approach in RE in Agile differs considerably from Waterfall method. For example, Cao and Ramesh [50] revealed that one of the key differences between Agile and Waterfall methods is that the former employs an iterative approach in discovering requirements to satisfy customers. Customers receive working features in shorter timeframes and are directly involved in modifying the requirements [25]. In contrast, RE activities are done upfront in Waterfall method and not re-visited throughout the project.

With respect to the prioritization of requirements, Agile and Waterfall make use of different techniques. In Agile, techniques are used to enable customers to provide information about the business value of features that meet their functional requirements in a particular iteration [25]. Hence, prioritization in Agile happens as part of the process of iteration [45]. For iterations, the requirements are ranked and maintained as part of a backlog list of user stories. In Waterfall, complete requirements for the entire project are prioritized based on features, functionalities, and non-functional related items upfront and the prioritization is maintained through the project lifecycle, and reprioritization is difficult [25].

Lastly, Agile and Waterfall methods differ in terms of their use of documentation. Agile methods are more code-oriented [28] and therefore requirement knowledge is not captured in comprehensive documentation. According to Chapin [12], the vast majority of Agile methods accomplish software maintenance without documentation. User stories do not become part of the complete documentation of the system [25] [18] and are only used to accommodate change at any time during the short development cycle. In contrast, plan-driven traditional methods such as Waterfall do not make such changes during the development cycle [18], and documentation is used to share knowledge [4] [31].

2.3 Software Quality Assurance

The objective of any software development process is to implement a quality product with no defects. Software Quality Assurance (SQA) is an integral part of this process [19] regardless of the software development method employed [21]. Sommerville [44] defines software quality as a management process concerned with ensuring the software has a small number of defects and that it reached the required standards of maintainability, reliability, portability and so on [44]. It also refers to the process of evaluating software to ensure compliance with software requirements [4] [43]. There have been many approaches proposed to improve software quality, including total quality management, six sigma, capability maturity model, and more [43]. Quality improvements help operational performance in several ways, such as increased revenue, reduced costs, and improved productivity [27]. SQA activities include auditing, testing of the software and an awareness checkpoint to examine the project's status [42]. According to Bohem (1984) [6], these activities are usually performed to determine the fitness or worth of a software product for its operational mission.

Due to the nature of software, it is challenging to assess the quality using the same set of guidelines for all products in all industries [3]. Many researchers have come up with standardized models that use particular dimensions, such as performance, reliability, responsiveness, aesthetics, durability, serviceability, and so on, to ascertain quality [43] [3]. The International Organization for Standards (ISO) proposed the ISO 9001 standard, which is applicable to a range of businesses that deal with software development [3] [43]. Although initially, quality was defined as conformance to a standard and specification, this definition was changed in the 1990s to keep up with the dynamic business environment. ISO 9126 was adopted

as the standard for evaluating software quality [43]. ISO/IEC 15504 is also known as SPICE (Software Process Improvement and the Capability determination) and refers to a “framework for the assessment of processes”. SPICE addresses the process, methods, and overall quality management, and also proposes standards for design, documentation, implementation, support, monitoring, control, ongoing review, and evaluation of current standards [3]. Different models such as CMM (Capability Maturity Model), CMMI (Capability Maturity Model Integration), TMM (Testing Maturity Model), TPI (Test Process Improvement) have been proposed to increase repeatability, predictability, and manageability of software [19].

2.3.1 Software Quality Assurance in Agile

Software Quality Assurance (SQA) in Agile is software testing that follows the same principles of Agile Software development method. Two of the most significant characteristics of Agile are 1) being able to handle volatile requirement and 2) being able to deliver in a shorter timeframe [32]. SQA in Agile also demands testing unstable requirements faster. In order to meet the demands, in Agile, the quality of software is maintained by testing and working with customer-approved software continuously throughout each iteration of the development cycle. Agile quality assurance is used to ensure that the software can respond to change, as the customer requires it. Since requirements in Agile are likely to change at any given time, these changes must also be tested. Huo et al. [32] identified that quality assurance starts at very early stages and continues during all stages (Figure 2).

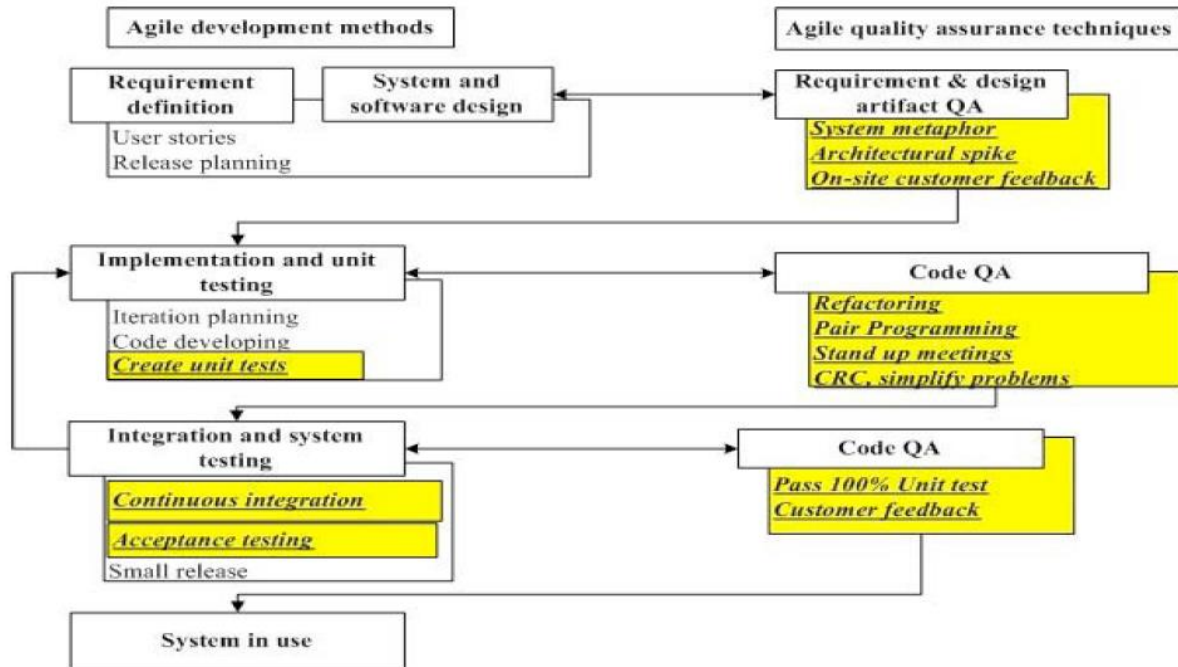


Figure 2: The relationship between Agile development methods and quality assurance techniques. [32]

In the above figure, it is notable that Agile quality assurance techniques take place at the same time as other processes such as requirement definition, user stories, release planning, and systems and software design. It also shows that these processes continue during the implementation and integration phases.

In order for the SQA testing to happen in a shorter time during any phase, all members of a cross-functional Agile team participate. All team members including Quality Assurance Analysts' are involved and encouraged to ensure the delivery of the business value desired by the customer at frequent intervals [1]. Agile quality is considered to be a result of practices such as effective collaborative work and communication during incremental development and iterative development. In this process, all team members including Quality Assurance Analysts' are

involved and encouraged to ensure the delivery of the business value desired by the customer at frequent intervals [1]. This means procedures of quality assurance in Agile Methods are dependent upon team collaboration [28], which serves as the focal point for ensuring quality [4] in a shorter time.

2.3.2 Software Quality Assurance in Waterfall

As mentioned earlier, the Waterfall model divides the software development lifecycle into linear stages with well-defined activities for each stage. The activities include 1) requirement definition 2) system and software design 3) implementation and unit testing 4) integration and system testing and 5) operation and maintenance [44]. Since the deliverables of one activity are required for the next activity, new stages will not begin until the previous phase is completed. In Waterfall, the traditional validation process includes activities that ensure that a software system is being built to the customer's satisfaction in the presence of a specification document [4] and results of the investigation are required as deliverables. In the testing stage, an investigation of whether the requirement provided by the stakeholders match the product outcome is performed. The testing stages of Waterfall method are done after development is complete [3] and its purpose is to determine the correctness of the product and the ability to move forward in the SQA process. If the software does not meet the requirements and produces errors, issues and defects are created, as a result. Although the documentation of issues and defects signals that the testing stage is complete, the development and testing stages are repeated until all issues and defects have been resolved. Figure 3 illustrates each stage and associated SQA activities in Waterfall model.

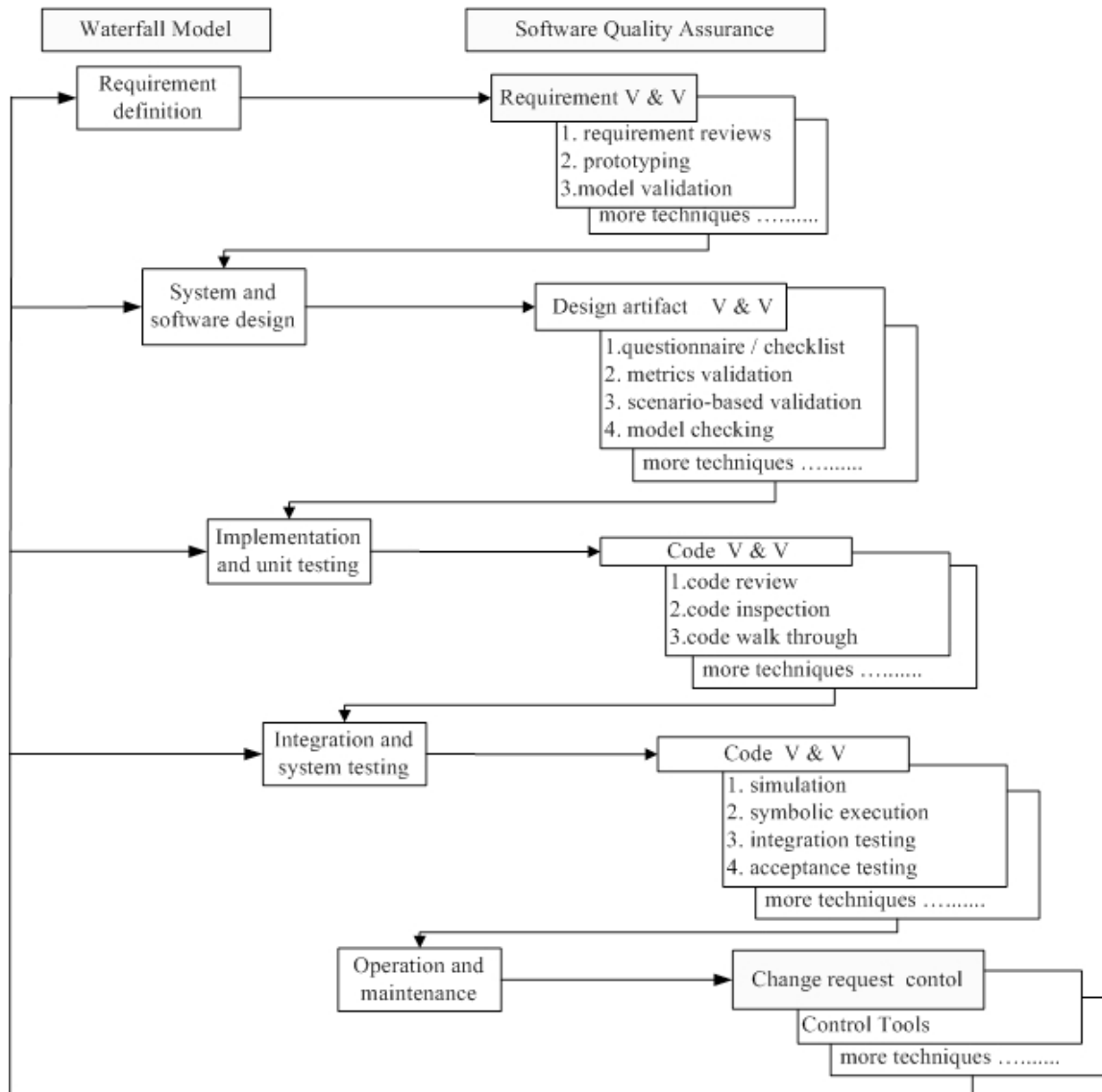


Figure 3: Stages and SQA activities in Waterfall [3]

2.3.3 Comparison of Software quality between Agile and Waterfall Method

The use of SQA in Agile and Waterfall method is different in terms of the initiation, frequency and focus of the activities. In Agile, SQA activities begin from the very beginning whereas, in Waterfall, SQA activities do not begin until all developments are completed. In the

field study conducted by Manjunath, Jagadeesh, and Yogeesh (2013) [30], the ability to detect issues earlier was noted to provide chances to fix them at an earlier stage.

A second difference between Agile and Waterfall method lies in how frequently the SQA activities are performed. According to Cao and Ramesh [22], early and constant validation and frequent review meetings are used to validate requirements in Agile. Also, Mnkandla and Dwolatzky [28] concluded that quality assurance occurs more frequently in Agile method than in Waterfall. Many metrics have been included, such as sprint wise issues, escapes per sprints, impediments count per sprint, and sprint health boards have been introduced to measure quality in Agile to accommodate its dynamic nature [1]. However, controversies exist with respect to the flexibility of these metrics to accommodate the changing requirements and whether the quality of the customers is satisfactory or not [21].

The focus of SQA activities is also different in Agile and Waterfall methods. The primary focus in Agile testing is to validate only the items for the particular iteration. In contrast, Waterfall method concentrates on the full requirements of the project. In Agile, unit testing and acceptance testing are the primary focus [22]. In general, there seems to be a lack of significance put in for Non-functional requirements (NFR) aside from functional requirements software solutions [13]. It is more evident in Agile than Waterfall. According to Taehoon et al., previous research with Agile approaches did not focus on non-functional aspects and suggested identifying non-functional requirements early can contribute to achieving improved quality [48].

2.4 Summary

The review of the relevant literature in this chapter presented the understanding of the software methods such as: Agile and Waterfall method with respect to the process of requirements and quality assurance. This chapter also introduced the comparison of requirement engineering and software quality assurance. The next chapter presents the research method used in this research project.

CHAPTER 3 METHODOLOGY

The purpose of the chapter is to present the approach used in this empirical study, as well as the rationale for choosing the case study method. The rationale for selecting the case study approach is described first, followed by a description of the various types of case study methods and the reasons for selecting an exploratory, quantitative design. The research procedures are presented in the next chapter (Chapter 4).

3.1 Rationale for Selecting Case Study Research

Empirical research is viewed as a rational approach to solving research problems using investigative techniques, designing alternative solutions, as well as validating, implementing, and evaluating the results [53]. Case study research methodology is viable in situations where individual, group, organizational and social phenomena are investigated [23]. It has the capability of examining a phenomenon in its natural setting and utilizing multiple data collection methods to gather information from one or more people, groups or organizations. Also, this type of research approach can address the “how” and “why” of a problem, as well as its complexity. Figure 4 shows the type of questions and problems that guide the design and execution of research studies [53].

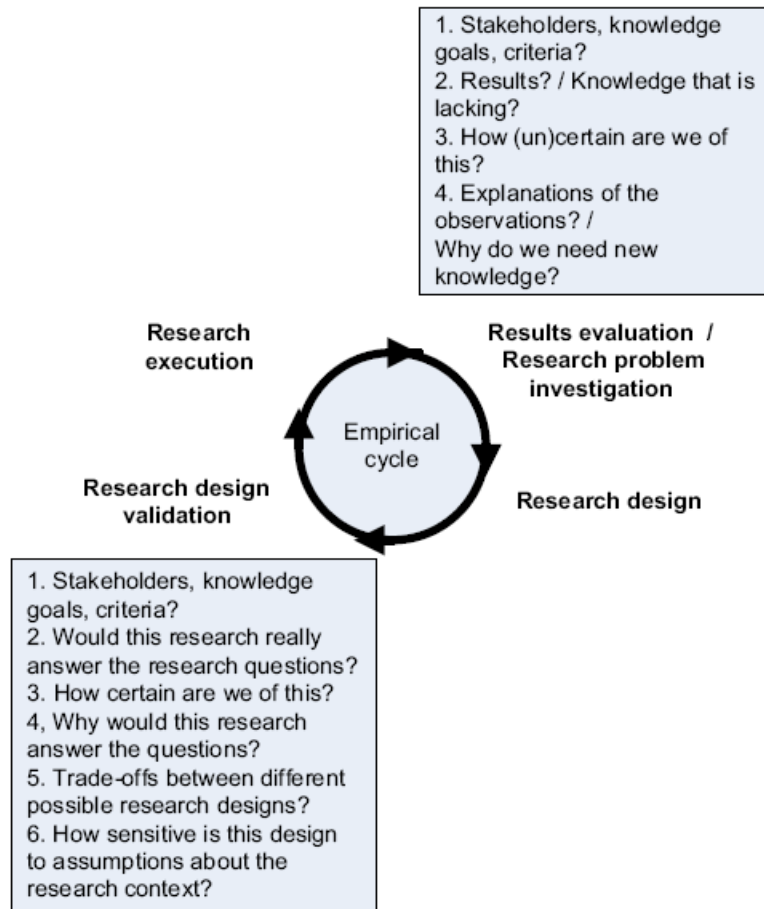


Figure 4: The Empirical Cycle of a Case Study [53]

3.2 Types of Case Study Research

Case studies can be accomplished in a single case or multiple cases [55]. Whereas a single case study concentrates on a single organization or situation, a multiple case study involves the examination of two or more instances of the phenomenon within the same study. Further, a case study can be categorized as exploratory, explanatory or descriptive in nature depending upon how it is implemented [55]. An exploratory case study is used to define the question and hypothesis for a subsequent study or to determine the feasibility of research procedures. The

purpose of an explanatory case study is to identify cause-effect relationships. In contrast, a descriptive case study first develops and documents the experience, and then answers a series of questions based on theoretical constructs [55]. Given the many options for investigation, there are at least six possible methods to employ in the case study approach (Table 1).

Type	Single-Case	Multiple-Case
Exploratory	Exploratory with Single-Case	Exploratory with Multiple-Case
Explanatory	Explanatory with Single-Case	Explanatory with Multiple-Case
Descriptive	Descriptive with Single-Case	Descriptive with Multiple-Case

Table 1: Types of Case Studies

In the present study, a multiple-case (completed for multiple organizations) with an exploratory method was selected. With respect to the selection of an organization, a business that specializes in transaction management for mortgage processing was chosen as the identified case. The primary reason for selecting the organization was because it provided an opportunity to examine two RE methods (e.g., Agile and Waterfall) that were employed at different times in the history of the company. This company had undergone a well-demarcated shift from Waterfall to Agile method that provided an opportunity to evaluate data from each period clearly

in terms of its outcomes, challenges, and benefits. The transition between methods occurred in 2012, and, therefore, all projects completed after 2012 followed Agile method, whereas pre-2012 projects followed Waterfall method.

3.3 Qualitative and Quantitative Research

Qualitative research methods are useful for exploring and understanding any emerging research problem. In this particular topic of Agile and Waterfall method, previous studies have employed qualitative research methods, such as field observation and interviews, to measure the impact of RE in Agile Methods on software quality [18]. Many of the qualitative studies that were completed helped gain an understanding of the complex underlying reasons, opinions, and motivations for using one method over another [25]. They investigated the why rather than the how many [4]. It also provided insights into the research problem for this thesis, and, in particular, helped in the formulation of ideas and hypotheses for this quantitative project. In contrast, a quantitative approach is used to quantify the problem by way of generating numerical data to transform into useable statistics and to evaluate and analyze the findings. In this approach, hypotheses were formulated for research problems/questions and relied on data, which were useful when evaluating cause and effect relationships.

Both qualitative and quantitative approaches have advantages and limitations. The decision of whether to choose a qualitative or quantitative is dependent on the nature of the project, the type of information needed, the context of the study and the availability of the data. There were a few reasons why a quantitative approach was chosen for this thesis. First of all,

while reviewing the literature, it was evident that there were many studies done to understand the characteristics of the Agile and Waterfall software development methods and their nature of requirements for quality assurance. However, no quantitative studies had been found that focused on the impact of adoption of Agile method on software quality.

In addition, a quantitative approach was selected due to the availability of data from an organization that had made use of both software methods (Agile and Waterfall) in recent years. Since this history had been documented, it provided an opportunity to measure and compare certain features of the completed projects. Statistical models could be constructed to test hypotheses about Agile and Waterfall method that might have usefulness in real life decision-making.

3.4 Summary

This chapter described the various types of case studies and research methods available for use in this project, as well as the rationale for choosing a case study approach to a quantitative research strategy. Based on the rationale, a case study methodology was considered well suited to compare the quality of projects completed under Agile and Waterfall method.

CHAPTER 4 RESEARCH METHODS AND DATA COLLECTION

This chapter presents the hypotheses of the research, data sources and case selection criteria, as well as the steps involved in collecting the data.

4.1 Hypotheses of the Research Project

The aim of this case study was to determine whether there is a difference between Agile and Waterfall methods. Two hypotheses for this project are described below:

Hypothesis 1: Number of defects in Agile and Waterfall projects is same.

$$\mu_{\text{Agile No. of Defects}} = \mu_{\text{Waterfall No. of Defects}}$$

Alternative Hypothesis: Number of defects is different in Agile and Waterfall projects.

$$\mu_{\text{Agile No. of Defects}} \neq \mu_{\text{Waterfall No. of Defects}}$$

[Where $\mu_{\text{Agile No. of Defects}}$ is population mean for a number of defects on Agile Projects and $\mu_{\text{Waterfall No. of Defects}}$ is the population mean for a number of defects on Waterfall Projects]

Hypothesis 2: Number of NFR-related defects in Agile and Waterfall projects is same.

$$\mu_{\text{Agile No. of NFR Defects}} = \mu_{\text{Waterfall No. of NFR Defects}}$$

Alternative Hypothesis: Number of NFR-related defects is different in Agile and Waterfall projects.

$$\mu_{\text{Agile No. of NFR Defects}} \neq \mu_{\text{Waterfall No. of NFR Defects}}$$

[Where $\mu_{\text{Agile No. of NFR Defects}}$ is population mean for a number of NFR-related defects on Agile Projects and $\mu_{\text{Waterfall No. of NFR Defects}}$ is the population mean for a number of NFR- related defects on Waterfall Projects]

4.2 Data Collection Strategy

This section describes the strategies of data sampling. In the first phase of the study, the objective was clearly defined and the data collection method was formalized. It was crucial to understanding what data was to be collected, from where and why it was required for testing the hypotheses.

4.3 The Organization

A financial organization with a medium-sized software team was selected as the source of data. This organization typically offers support to financial institutions such as banks and mortgage lenders, and has a software team known to provide technological solutions to a diverse range of clients over a number of years under Waterfall and Agile methods. This particular team worked in an industry that is both fast-paced and dynamic, and so, this selected company was perceived to be a good representation of a medium-sized Information Technology organization.

4.4 Formation of the Agile and Waterfall Group

A method was needed to identify and form two distinct groups (e.g., Agile, Waterfall) for the purpose of testing the hypotheses. Since, the software team adopted the Agile method in 2012, any projects that started after 2012 were assigned to the Agile group. Projects that were initiated before 2012 were assigned to the Waterfall group. The year of initiation of a project was used to assign a project to each group as shown in Table 2.

YEAR PROJECT WAS INITIATED	ASSIGNMENT TO GROUP (AGILE OR WATERFALL)
2014	Agile
2013	Agile
2012	Agile
2011	Waterfall
2010	Waterfall
2009	Waterfall

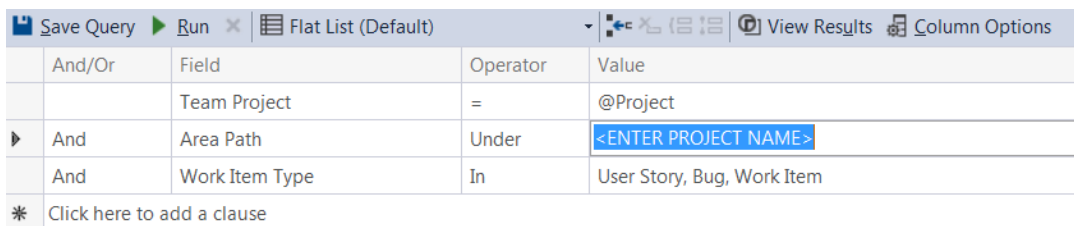
Table 2: Relationship between Year of Project Completion and Group Assignment

Thirty projects completed by the software development team were randomly selected from the years 2010 to 2014, with 15 projects from the years 2012-2014, and 15 projects from the years 2010-2011, which corresponded to Agile and Waterfall methods, respectively.

4.5 The Source of Data

Once the groups had been formed, additional information had to be retrieved. An application lifecycle management system, called Team Foundation Server (TFS), was used to track and store the information for this company. TFS is a product that provides source code management, reporting, requirements management, and project management for Agile and Waterfall methods. This system not only kept a detailed history of all projects including the number of change requests (combination of user stories or work items and defects), but also held hours spent and the final result of testing each change request. It also kept track of the number of defects created for each change request.

For the purpose of this research, the data from TFS was accessed and collected using several steps. First, the number of requests/changes was captured to understand the size of a project. Then, the time and effort put into each request was also obtained to come up with the duration of the project. In this process, the name and number of people involved in completing the requests were also recorded. Lastly, the number of defects found in testing and their nature of the problems were also collected. By creating an appropriate query and report from Team Foundation System with the following: ‘Number of Change Request’, ‘Quality Testing Result’, and ‘Number of Defects’ were generated. Figure 5 shows the query used to collect data from TFS.



And/Or	Field	Operator	Value
	Team Project	=	@Project
▶ And	Area Path	Under	<ENTER PROJECT NAME>
And	Work Item Type	In	User Story, Bug, Work Item
* Click here to add a clause			

Figure 5: Generic query used to execute the report from TFS

The purpose of ‘Number of Change Request’ was used to determine the size of the projects and to compare which took longer to complete. ‘Number of Defects (bugs)’ was used to evaluate which projects had produced better quality where better quality was associated with less number of defects. Table 3 shows a sample report of defects from TFS.

The selected projects enabled comparisons to be made between Agile and Waterfall methods to understand the overall impact. Next, data fields had to be selected for analysis; it was also important to comprehend the nature of the data that were available for the project first.

ID	Type	Hours Spent	Title	Found During	Scrum Team
25546	Defect	7.5	<Title 1>	Dev/Functional Testing	<Team Name>
25616	Defect	5	<Title 2>	Stage/Internal Acceptance Testing	<Team Name>
26683	Defect	3	<Title 3>	Prod/Deployment	<Team Name>
26688	Defect	3.25	<Title 4>	QA/Deployment	<Team Name>
25545	Defect	5.75	<Title 5>	Dev/Functional Testing	<Team Name>
25547	Defect	4.5	<Title 6>	Dev/Functional Testing	<Team Name>
25615	Defect	3.5	<Title 7>	Stage/Internal Acceptance Testing	<Team Name>

26676	Defect	1	<Title 8>	QA/Deployment	<Team Name>
-------	--------	---	-----------	---------------	-------------

Table 3: Sample report of defects in a project

4.6 Criteria for Case Selection:

In order to select the cases, the characteristics of the project and teams were significant. According to the survey done by Vijayasathy and Butler [51], it is evident that organizations with methods such as: Agile, Traditional, Iterative, and Hybrid exhibit different characteristics in terms of team size, project size, revenues and project criticality. Since the purpose of the project was to compare the quality of Agile and Waterfall methods, three selection criteria were used to standardize effort or reduce bias in each of the two groups. These criteria were: project size, team size, and duration of the project.

4.6.1 Project Size:

First, the size of a project was the most important factor when selecting projects for inclusion in the analysis. The size of the project was determined using the number of work items or the number of user stories for each project. Work items and user stories represent the number of features/functionalities that a project required throughout the project. A detailed explanation of user stories and work items is presented below:

4.6.2 User Stories:

Each project under Agile method was reviewed by the number of requests, called user stories, in which software requirements are documented in a less formal way as the primary unit [4] [30]. A user story is structured in concise format that describes in natural language the “Why” and “How” of a project [17]. The user stories are typically written in the form of “As a (Role), I want (Something) so that I can (Benefit)”. To complement the informal statement of the user story, acceptance criteria are also included in order for the software quality to be considered fulfilled. ‘Acceptance Criteria’ are written in ‘Given,’ ‘When’ and ‘Then’ format (Table 4).

Title	Approve New User Request
User Story	As a 'User Admin' role, I want to approve pending user request so that new users can access the system.
Acceptance Criteria	Given I want to access pending user request When I login to the system and access 'Pending User Requests.' Then I view a list of users with "Approve" and "Decline" option. Given I want to approve pending user request When I am on 'Pending User Requests' screen, and I click on "Approve" link Then I view confirmation message showing "The user has been granted access to login."

Table 4: Sample format of a user story and acceptance criteria

4.6.3 Work Items:

In projects completed under Waterfall method, a traditional format was presented that included a Business Requirement Specification Document (BRSD). The BRSD is a written plan completed before beginning software design and implementation phase. Requirements in the specification documents are broken down into change requests called 'Work Items' that are primarily based on system feature functionality. Unlike user stories, work items are not in any consistent format, but they are broken down by feature and functionalities. In some cases, one work item can be translated into multiple user stories, whereas, in other instances, one user story can be broken down into multiple work items. In the end, requests in the form of a user story represent how a feature will need to be used from a user perspective, and work items represent how a feature will be used, in general.

After collecting the user stories and work items for a project, dispersion graphs were used to select cases to analyze projects in the Agile and Waterfall group (Figure 6 and Figure 7).

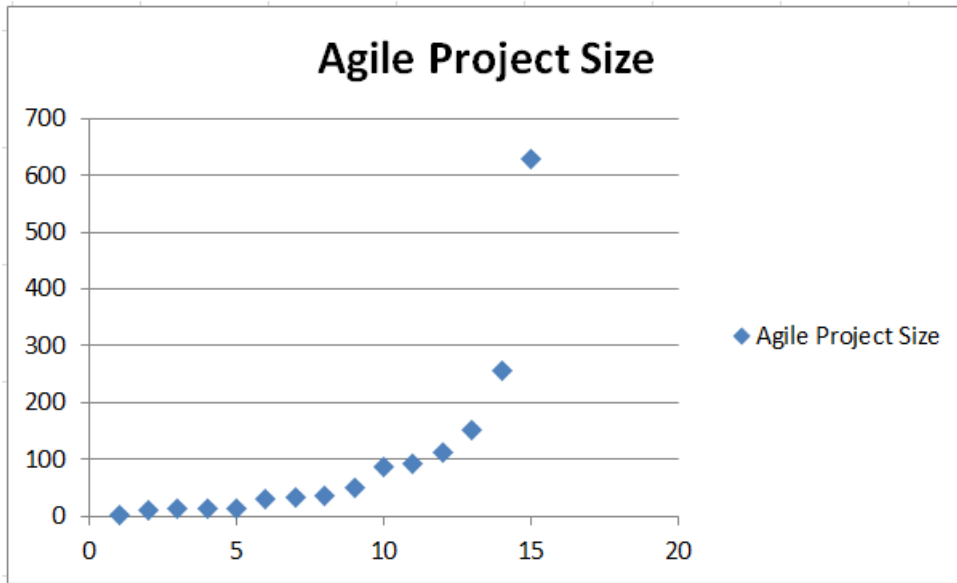


Figure 6: Size of selected projects using an Agile method

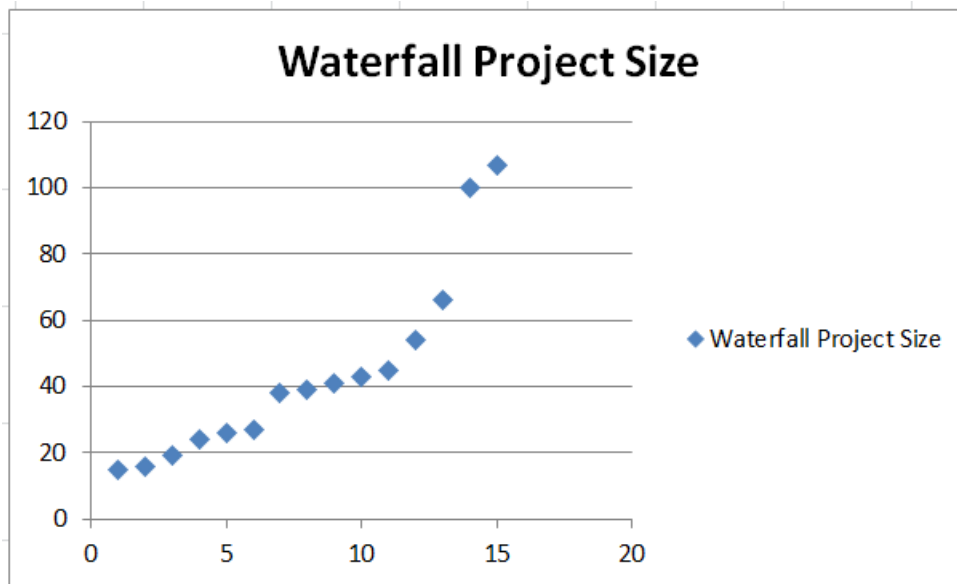


Figure 7: Size of selected projects using an Agile method

4.6.4 Team Size:

Secondly, it was determined that team size may be a factor that could bias the comparison between Agile and Waterfall methods, and, therefore, dispersion measures of this variable were also taken into consideration when selecting cases. Upon inspection, it was determined that the software team consisted of 35-40 members, which were broken down into smaller sub-teams to complete various projects. For example, for a medium sized project, a team typically consisted of a project manager, a product owner, 2-6 developers, and 1 quality assurance analyst. It was observed that some projects had small teams comprised of 2-4 members while other projects had moderate team size (5-11 members), and the rest had large team size (12-18 members). Thus, the goal was to select cases that had similar sized teams given that success with Agile method is heavily dependent on team collaboration and communication. A moderate team size was given priority for selecting the projects.

4.6.5 Duration:

Lastly, project duration was considered an important factor in case selection. Longer durations for a small sized project suggest that more time was permitted to comprehend and complete tasks relative to a project completed within shorter timeframes. Duration was calculated by examining the time that elapsed between project initiation and project completion. Within the 15 randomly selected projects in both Agile and Waterfall group, duration ranged from 1 month to 2 years. Projects that had a duration of 3-9 months were considered suitable for inclusion, and cases with a smaller duration (1-2 months) and longer durations (10-24 months) were excluded. Figures 7A and 7 illustrate the similar duration of the projects in the Agile and Waterfall groups. It is also notable that the consistency in the team size and duration across projects, with greater variability noted in the project size.

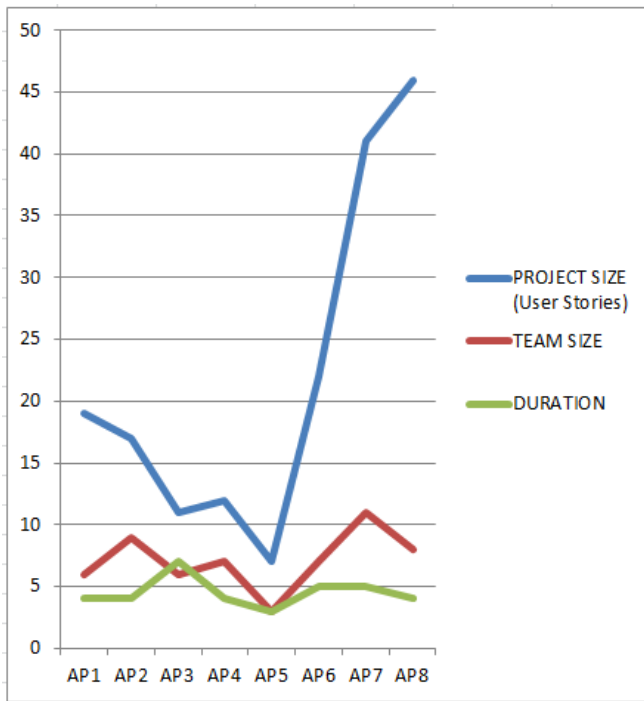


Figure 8: Team size, project duration and project size for each Agile project selected for analysis

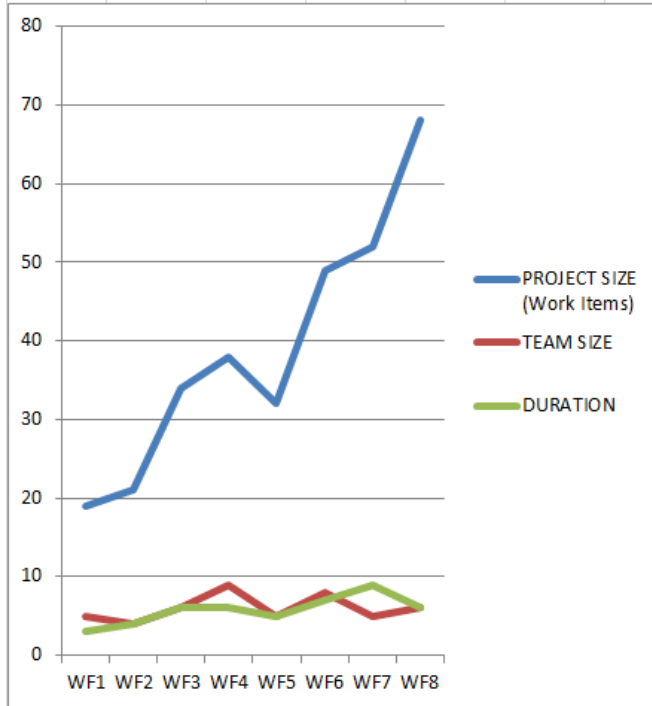


Figure 9: Team size, project duration and project size for each Waterfall project selected for analysis

4.7 Description of Agile and Waterfall Projects

Tables 5 and 6 describe selected Agile and Waterfall projects for this research:

No.	Description
AP1	For this project, team of 6 had completed 19 requests in 4 months in-order for a client to access and monitors their status via email notifications.
AP2	This project was initiated to allow employees to self-register to a web portal and to access and retrieve mortgage documents for their clients. It enabled managers to approve employees who requested access to the web portal. This project involved 9 people (including 1 project manager, 1 product owner, 1 quality assurance analyst, 1 development team lead, 4 developers, and 1 user acceptance analyst) and lasted for 4 months. It was broken into 17 items throughout the process.
AP3	The purpose of this project was to improve a specific process of a financial company, including enhanced traceability of deals and automated reporting. This project started in 2013 and lasted for 7 months. Six team members (including 1 project manager, 1 product owner, 1 quality assurance analyst, 1 development team lead, 1 developer, and 1 user acceptance analyst) were involved in the project for a size of 11 user stories.
AP4	For this project, changes were made to existing software that enabled the ability to submit multiple orders and cancel orders. In only 4 months, some maintenance work (broken into 12 user stories) of an existing application was completed.

AP5	This project was a small portion of a large project that had many deliverables. Only a few items were analyzed and scheduled under this sub-project to deal with 3rd party application integration. The project was completed in over 3 months and started in late 2013. Five team members (1 product owner, 1 developer, and 1 quality assurance analyst) completed the project size of 7.
AP6	This project was initiated following a request from an existing client. It involved modifying an existing web application to include a brand new role with new features and functionalities. The team included 7 members (1 project manager, 1 product owner, 3 developers and 1 quality assurance analyst), and 22 user stories were completed over 5 months.
AP7	This project involved launching a new product for a client. The primary focus was to establish communication between systems and to develop/enhance system solutions that process and monitor mortgage-related transactions on a web portal. This project was completed starting in 2013 for 5 months and involved 11 team members for 41 user stories.
AP8	This was the largest project in the group with 46 requests and completed by a team of 8 in the span of 4 months. This allowed a mortgage lending institution to retrieve data and risk factors from the internal processing system.

Table 5: Description of Agile projects

No.	Description
WF1	This project was completed to setup a marketing portal for sending and retrieving marketing communication on an existing web portal. A team of 5 completed this project (19 work items) within 3 months.

WF2	<p>This project was to provide application features that offered cost-effective programs for flow-through mortgage processing and appraisal management. It was completed in 2010 over 4 months with the team of 4, including 1 business analyst, 2 developers and 1 quality assurance analyst with total of 21 work items.</p>
WF3	<p>This project involved the introduction and implementation of a new web portal where existing and new clients could submit instructions and monitor file progress to ensure consistency and efficiency. This project was completed with 34 items over 6 months. A team of 6 people, including 1 project manager, 1 business analyst, 3 developers, and 1 quality assurance analyst, completed the project.</p>
WF4	<p>This project involved the development of a web application that enabled submission, tracking, and reporting on real-estate appraisal transactions via an online web Portal, as well as the ability to track ‘refinance’ transactions. For this project, a team of 9 was used, including 1 business analyst, 6 developers and 1 quality assurance analyst were involved to tackle 38 items within one month.</p>
WF5	<p>This project enhanced existing program features to give the capability to manage the instructing, funding and reporting of ‘Flow Through’ Purchase and Refinance requests for a financial institution. This project had a team of 5 people consisting 1 business analyst, 2 developers and 1 quality assurance analyst to work over 5 months to complete 32 items.</p>
WF6	<p>This project involved the construction of an application that enabled clients to place orders for Appraisal services, view/track status of Appraisal and retrieve the output of the appraisal process via web portal. This project was completed over 7 months with 8 team members that consisted of 1 project manager, 1 business analyst, 4 developers, and 2 quality assurance analyst. The size of this project involved 49 items, which was the largest of the selected six waterfall projects.</p>

WF7	This project was initiated to enhance cost savings by streamlining the process for mortgage-related transactions for a client. The focus of the project was to reduce the number of manual steps involved in final reporting and reconciliation of files through a reduction in the printing o and delivery of documents. The size of this project was the second largest of the six waterfall cases selected for analysis (52 items). Five people were involved completing the project in a 9 month period.
WF8	WF8 project was the largest with 68 requests, completed by a team of 6 over 6 months. The purpose of this project was for external accounting software to integrate with internal software.

Table 6: Description of Waterfall projects

4.8 Outcome Variables:

In order to analyze the data and test the hypotheses, the following outcome variables were selected based on the particular interests of this research project.

4.8.1 Defects:

The variation between expected and actual results during software quality assurance phase is known as defects. Different organizations have different names to describe this variation, but commonly used terms to describe defects include bugs, problems, incidents or

issues. All the defects considered were those identified after the deployment of the software and not those during the development of the software

4.8.2 Number defects:

This is the number of defects that were found during the quality assurance phase of the project and was a measure of quality in the project. More defects meant more issues were found and needed to be corrected before the project could be considered completed. For the purpose of this research, only requirement related defects were considered regardless of their severity. These defects were then categorized into 2 groups: Non- Functional Requirement (NFR) related defects and Functional Requirement (FR) related defects.

4.8.2.1 Number of NFR related defects:

Defects put into NFR related defects category was based on their characteristics.

4.8.2.2 Number of FR Related defects:

Just as mentioned above, FR related defects were also categorized based on the nature of the defects.

4.9 Threats to Validity

This section focuses on the general validity of the case study research and whether the effects observed in this study are due to the manipulation of the independent variable and not some other factor.

4.9.1 Threats to Internal Validity

This section focuses on the general validity of the case study research. *Biased subject selection*, as a threat to internal validity was controlled by carefully selecting projects based on the selection criteria: project size, team size and duration. *History* as a threat to internal validity was minimized by selecting projects from recent years. *Instrumentation* as having the projects controls a threat to internal validity conducted by the same team and in similar settings. Other threats to internal validity such as Experimental mortality, statistical regression, maturation and testing are not applicable to this research.

4.9.2 Threats to External Validity

Interactions between selection biases and the independent variable were controlled due to the initial random sample selection. Reactive testing, reactive effects of experimental arrangements and multiple treatment interference threats are not be applicable to this research because there is be no pretesting involved that may have affected reactions to experimental variables.

The external threats regard the ability to generalize the results outside of the experiment setting. Firstly, what would be the difference if this experiment were performed with other projects? Further, what would be the results if they were done for different industry? Also, what would the results be for much larger sample size? To ensure this research design is valid, type of defects of any severity considered for the project was carefully reviewed. Only requirement related defects were considered and these were reported at the end during user acceptance testing.

4.10 Statistical Approach

The study objective was to examine the difference between Agile and Waterfall method with respect to software quality. In order to analyze the data, descriptive statistics were calculated (mean, standard deviation and standard error of the mean) for the following variables: Project Size (Number of User stories or Work items) and Number of Defects in each of the two group (Agile and Waterfall). Statistical testing was used to determine the significance of the difference between the two groups. A Mann-Whitney U test was performed on two independent groups (Agile and Waterfall) for each outcome variable to compare the difference between means. For the test, a typical alpha level was chosen ($\alpha = 0.05$). All statistical analyses were completed using IBM SPSS Statistics Version 22.

4.11 Summary

This chapter discussed the process of case selection, as well the procedures for data collection and analysis. Data were collected and compiled based on several selection criteria. A quantitative approach was needed in order to prepare the data for statistical analysis of the Agile and Waterfall group.

CHAPTER 5 RESULTS AND DISCUSSIONS

The goal of the analysis was to determine whether the quality of projects differed between Agile and Waterfall methods. This chapter describes the empirical findings of the data collection and analysis process. A within-group analysis was performed first to evaluate the project size and effort in each of the two groups. Then, this work examined whether Waterfall and Agile software methods differed with respect to quality, using the number of defects as a measure of quality.

Motivation to center this study on the number of defects was driven by the contradictory behaviour regarding documentation when comparing Agile and Waterfall methods. To keep up with today's fast changing needs of the clients; Agile methods promote faster project delivery in a shorter timeframe and allow for changes at any time. However, in this process, assuring the quality of software becomes more challenging without knowing and understanding the requirements. Since Agile has almost zero documentation, data related to the length of activities and budget implications were considered hard, if not impossible, to compare and should be dealt with separately. Hence, quality comparison from the perspective of duration of the project and budget implications was left to be carried out on future work. Lastly, after discussing the results in terms of other studies that have made similar comparisons, the strength and limitations of this project are presented.

5.1 Analysis of User Stories and Defects in Agile Group

The mean of all Agile projects' user stories was 21.9 (SD 14.2) and mean of the defects was 12.5 (SD 4.9). Figure 10 illustrates the proportion of user stories and defects for each of the Agile projects. Defects comprised between approx. 30%-60% compared to the user stories in the Agile group.

Further analysis was conducted on the defects related to non-functional and functional requirements. In the Agile group, the mean number of NFR-related defects was 8.6 (SD 3.8), whereas 3.9 (SD 1.7) defects were FR-related. Figure 11 illustrates the proportion of NFR and FR defects in each Agile project. It shows that more than 60% were NFR related defects in this group.

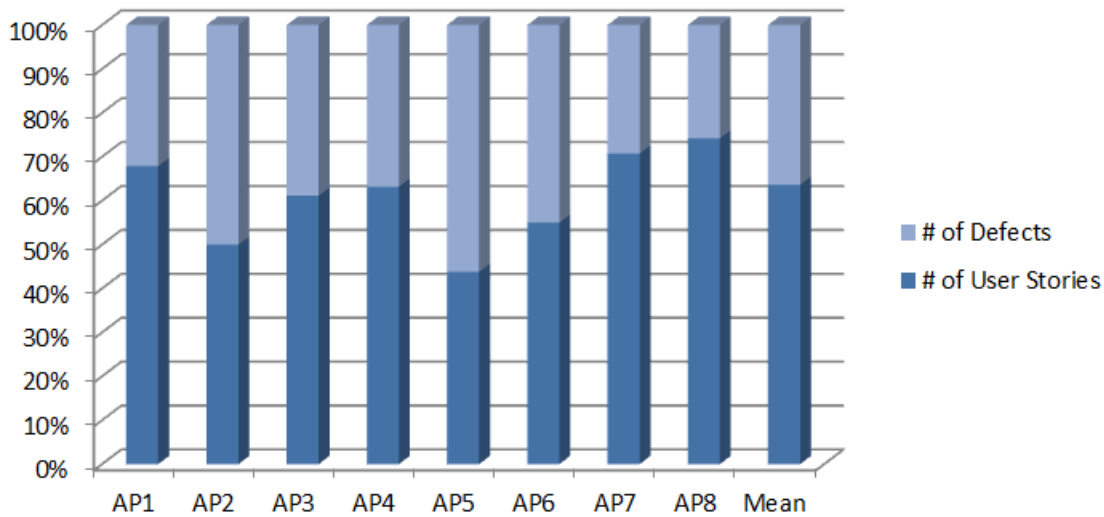


Figure 10: Agile Projects: Proportion of User Stories and Defects

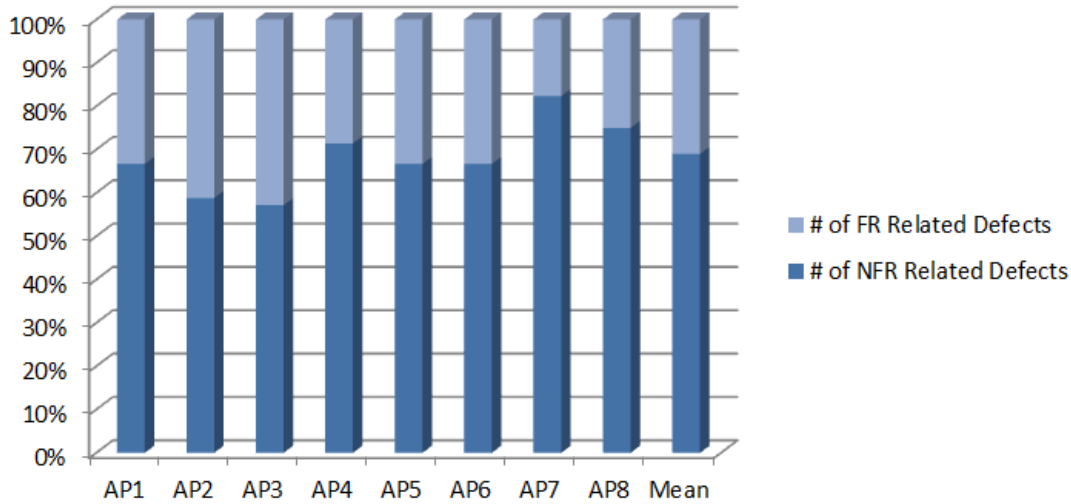


Figure 11: Agile Projects: Defects with proportion of NFR and FR related defects

5.2 Analysis of Work Items and Defects in Waterfall projects:

The mean of all work items in Waterfall projects was 39.1 (SD 16.5) work items and 8 (SD 4.2) defects. Figure 12 shows the proportion of defects in each of the Waterfall projects, which fell in the range of 8-33% compared to the user stories and work items.

Next, when the defects were analyzed and grouped into non-functional and functional requirements related categories, in the Waterfall group, the mean number of NFR-related defects was 4.1 (SD 1.7), whereas 3.9 (SD 3.7) defects were FR-related. Figure 13 illustrate the proportion of NFR and FR defects in each Waterfall project. This figure shows that a couple of projects did not have any NFR related defects, but on average, more than 50% of the defects were made up of NFR related defects.

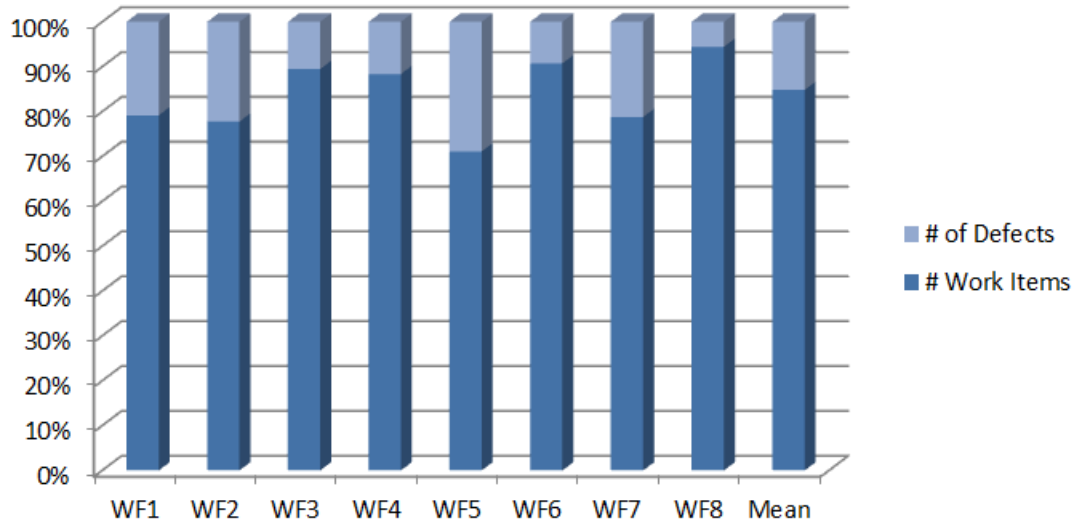


Figure 12: Waterfall projects: Proportion of work items and defects

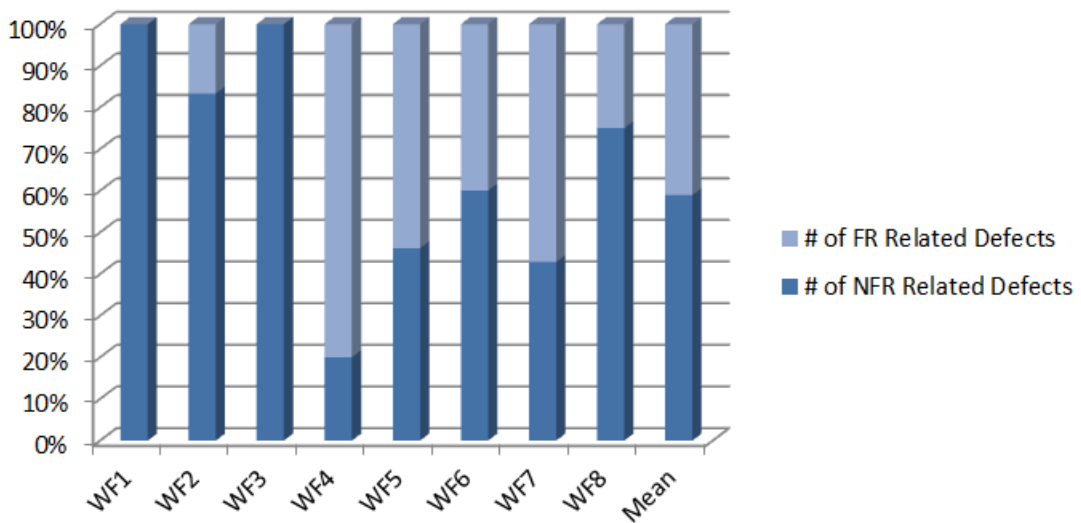


Figure 13: Waterfall projects: Defects proportion of NFR and FR related defects

5.3 Comparison between Agile and Waterfall Projects

This section compares the projects from Agile and Waterfall group in terms of number of defects and examines the hypotheses. Table below shows the group statistics of both Agile and Waterfall group.

	Group	Mean	Std. Deviation	Std. Error Mean
Project Size	Agile	21.9	14.2	5.0
	Waterfall	39.1	16.5	5.8
Number of Defects	Agile	12.5	4.9	1.7
	Waterfall	8.0	4.2	1.5
Number of NFR Defects	Agile	8.6	3.8	1.3
	Waterfall	4.1	1.7	0.6
Number of FR Defects	Agile	3.9	1.7	0.6
	Waterfall	3.9	3.7	1.3

Table 7: Group Statistics of Agile and Waterfall

5.3.1 Hypothesis 1:

There was a significant difference in the number of defects for Agile (M=12.5, SD=4.9) and Waterfall (M=8, SD=4.2) (Mann-Whitney U=12, $n_1 = n_2 = 8$, $P < 0.05$ two tailed). Projects using Agile method had significantly more defects than projects using Waterfall method. The alternative hypothesis was accepted in this case, and results showed that projects in Agile produced higher number of defects. Below figure shows the both means of number for defects for Agile and Waterfall side by side.

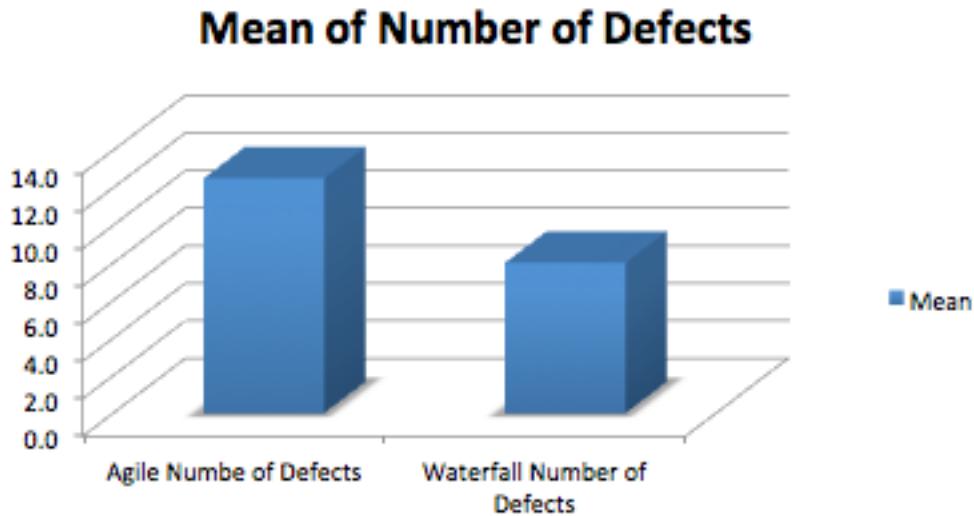


Figure 14: Mean of Number of defects for Agile and Waterfall

To date, only one prior study could be found showing that Waterfall method provided better quality than Agile method. The Standish Group, which has been collecting case information on real-life IT environments and software development projects since 1985, examined project success and failure in more than 90,000 IT projects reported that, of the projects completed between 2003 and 2012, 48% of Agile projects were ‘challenged’ compared to 43% of Waterfall projects. ‘Challenged’ projects were defined as those that were delivered late, over budget, and/or with less than the required features and functions [46].

In contrast, some studies have suggested that Agile methods have advantages over Waterfall method. For example, in an analysis [32], Ming et al. evaluated the differences in quality between Waterfall and Agile methods under conditions of time pressure and an unstable requirements environment. They concluded that the SQA abilities, frequency and the time of implementation could contribute to the success of the Agile method under these conditions.

They also acknowledge that comparing the quality resulting from the use of Waterfall and Agile is difficult due to the difference in initial development conditions and costs [32].

In another case study [26] that compared two releases of the same product completed by the same personnel revealed that 65% improvement in pre-release quality and a 35% improvement in the post-release quality was noted in methods using XP compared to Waterfall. Though the number of defects was used as a measure of quality, the size of the new release was one fifth of the old release that was compared to this research. Significantly smaller release may have contributed towards quality improvement of XP over Waterfall. Hence, findings from this study suggest that when using a more inclusive and less biased sample, Agile will deliver projects with a significant larger number of defects than waterfall.

Other studies regarding the quality of Agile and Waterfall methods have not shown the superiority of one method over another. No reliable result was found for quality in the study done by Benediktsson et al. to investigate the impact of software development approach [10]. F. Macias (2004) reported no significant difference when comparing XP and traditional software development (pilot study completed by 2nd year undergraduate students) approaches in terms of quality and size of the product the time required to produce product [29].

5.3.2 Hypothesis 2:

The number of NFR related defects for Agile (M=8.6, SD=3.8) was significantly higher than the Waterfall (M=4.1, SD=1.7) (Mann-Whitney $U=9.5$, $n_1 = n_2 = 8$, $P < 0.05$ two tailed). Agile had more NFR-related defects than Waterfall method that suggested that the non-functional

requirements may not be as clearly understood upfront in Agile Method. Hence, the alternative hypothesis was selected. In fact, from the literature review, no significant approach could be identified where NFRs were dealt with explicitly. Below figure shows the both means of number for NFR related defects for Agile and Waterfall side by side.

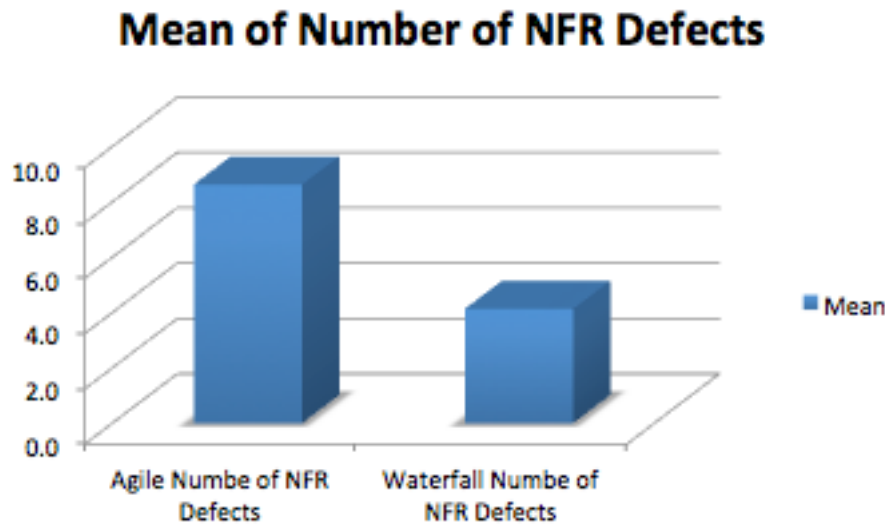


Figure 15: Mean of NFR-related Number of defects for Agile and Waterfall

Many works support that Agile method deal less with NFRs compared to Waterfall [6][13][14][15][48]. One of the suggested reasons for this is that NFRs are not always apparent when dealing with requirements in increments having functionality as the focus. Although the ISO 9126 series provided quality measurement metrics and various quality evaluation guidelines for a general software project, these approaches were on the basis of well-defined documents, which is not present in Agile [48]. Also, Agile have not adequately modeled Non-Functional Requirements (NFRs) and their potential solutions (operationalization) in early development phases [14]. However, Farid has supported the idea that Agile processes can still benefit from

capturing NFRs as first-class artifacts early on and not treating them as an afterthought process. He proposed visualization tools to use to develop the modeling NFR in Agile.

Though there have not been any empirical research studies that directly compared NFR defects in various software development methods, Kassab found from his empirical study [24] that 68% of Agile projects reported the use of NFR during estimation compared to 40% for Waterfall. These data suggest that Agile method involve more NFRs than might be expected from current understanding when compared to Waterfall method. However, the results obtained in this study, as shown above, seem to contradict the findings of Kassab's work. One possible explanation may be that even if Agile pays more attention to NFR while estimating size/effort as indicated in [24], Agile projects might indeed not deal with NFR as well as waterfall projects do. Possible reasons for that could be a) The fast pace of the Agile method do not meet the need for carefully eliciting and modelling NFR. b) There are a few approaches to deal with NFR [13] [11] [28] [53] that could be applied to Waterfall method, but would be difficult to be used in Agile method. c) The fast incremental pace together with the minimal documentation may severely impact NFR elicitation and modelling.

The findings of this work support the need for continued attention to understanding, documenting and communicating requirements in the software development process. As mentioned in earlier chapters, the documentation of comprehensive and complete requirements is present at the outset of a Waterfall project, but in Agile, the requirements are communicated in increments and in a less formal way. Though Agile methods have gained popularity, many practitioners and researchers are unclear about how the requirements are dealt with in Agile [31] and doubt the benefits [21]. Although Paetsch [32] suggested that using practices that improve the adoption of RE in Agile projects is important, no evidence had been provided to demonstrate

that these suggestions have been tested in actual projects. The findings from this study show that the method that employed complete and stable requirements at the outset of a project (i.e., Waterfall) produced fewer requirement-related defects, and thus, supports the value of understanding requirements early in the software development process.

5.4 Strengths

Although there are general studies on Requirement Engineering (RE) and Software Quality Assurance (SQA) of Agile, there are only a few empirical studies conducted to understand the impact RE has on SQA in Agile. Therefore, these findings contribute to the impact of RE on SQA in Agile.

First, one of the strengths of this work lies in its examination of materials and the real-world (i.e., ecological validity). This has practical applications that are relevant and useful to life. This type of research is hard to duplicate in a controlled environment with limited resources. The evidence suggests that the process of adopting Agile need to be well thought out in ways that enable more attention to requirements.

Secondly, this study accessed a combination of projects that catered to different clients in the financial industry. While there are research studies available in other sectors, none has been performed in a similar organization. Also, having different clients for each project offers feedback from multiple sources.

Lastly, since the findings of this work suggest that NFR related issues were responsible for Agile presenting a larger number of defects than Waterfall, companies that have adopted or are considering to adopt Agile method should consider NFR as a mandatory part of RE. Otherwise,

savings due to faster development process and lower budget may end up being reversed after deployment as a result of more time and money spent to fix defects that could have been avoided.

5.5 Limitations

The aim of this research project was to examine empirically whether Agile method produces a better quality product. The case study method was chosen so that the information could be useful in similar settings to better understand which of the two methods, Agile, and Waterfall, might be adopted. Limitations associated with the case study are discussed in this section.

The first limitation is related to the selection of the projects. As it may happen in most of the situations where such a study may be carried out, the developers were not the same for all the projects. However, these developers belonged to smaller sub-teams, which were part of a one big team under the same management. This fact may mitigate any selection bias considerations.

Next, data collection regarding the number of defects might have a little bias towards Agile method. Unlikely in the Waterfall process, during the Agile projects some of the defects may not even be registered as such. They might have been detected during an interaction to solve another defect and since Agile has a policy of documenting as less as possible, no annotation regarding this new defect would have been made. However, from observing and participating in similar projects in the same organization (not used in this study), it is possible to state that the numbers

of defects that might have been not logged would not be significant enough to alter the results. In fact, it could only enforce the results obtained in this work.

In this chapter, the summary of the key findings of this work and the results from other similar studies are discussed. Comparing the results from similar studies and reviewing the strengths and weaknesses, it was evident that the influence of lack of requirements and inadequate process for including NFR in Agile has resulted in poor quality in this empirical study.

5.6 Summary

This chapter presented the results of the statistical testing of the data collected. Also, the summary of findings are discussed in terms of what can be learned from the introduction of Agile methods into other similar organizations in the information technology industry, and possible future opportunities are presented for further research.

CHAPTER 6 CONCLUSION AND FUTURE WORK

This study compared the impact of software development methods on quality using software development projects worked on by a company comprised of several sub-teams that provide innovative solutions for mortgage and loan related services to Canadian Financial Institutions. The company switched from Waterfall to Agile method in 2012. Thirty projects from this organization were initially chosen. To minimize selection bias, medium sized projects (i.e., 7-68 user stories or work items) that were worked on by medium sized teams (i.e., 3-11 team members), and spanned between 3 and 9 months in length were identified as project for inclusion. Two significant findings were noted. First, Agile projects had significantly greater number of defects. Second, when the defects were categorized into non-functional and functional requirements, comparisons between the two groups revealed that Agile projects had more NFR-related defects than Waterfall expressively.

The goal of this thesis is not to advocate one method or another. The main goal is to use data extracted from several real life projects to contribute to understanding if the adoption of the Agile method impact quality negatively or not. It is important to note that since considerations regarding budget and time are hard to be generalized, the focus was put on the number of defects as a more stable parameter to be measured. These results suggest that choosing Agile over Waterfall may lead to a larger number of defects and, therefore, to lessen the quality possibly due to the lack of understanding of the requirements. This study contributes to other company's decision-making process when choosing what method to adapt and apply for them. Though some perceive that the Agile process may be the universal remedy for software development

project failure [46], this research supports that; agility without effective requirements cannot bring success [35].

As future work, studies using larger sample size will be performed, which may offer findings that can help to confirm current findings. Another possible future work is to collect data from various organizations based on company size, team size, and different type of industry.

REFERENCE

- [1] Agarwal, A.; Garg, N.K.; Jain, A., "Quality assurance for Product development using Agile," 2014 International Conference on Optimization, Reliability, and Information Technology (ICROIT), pp.44,47, 6-8 Feb. 2014
doi: 10.1109/ICROIT.2014.6798281"
- [2] Agile Alliance, Manifesto for Agile Software Development, 2001.
[Http://www.agilemanifesto.org/](http://www.agilemanifesto.org/)
- [3] Alsultanny, Y.A.; Wohaishi, A.M., "Requirements for Software Quality Assurance Model," Second International Conference on Environmental and Computer Science, 2009. ICECS '09., pp.19-23, 28-30 Dec. 2009 doi: 10.1109/ICECS.2009.43
- [4] Araujo, J.; Ribeiro, J.C., "Towards an aspect-oriented agile requirements approach", Eighth International Workshop on Principles of Software Evolution, 5-6 Sept. 2005, pp.140-143
doi: 10.1109/IWPSE.2005.31
- [5] B. Boehm, "Requirements That Handle Ikiwisi, COTS, and Rapid Change," Computer, July 2000, pp. 99–102.
- [6] B. W. Boehm, "Verifying and Validating Software Requirements and Design Specifications," IEEE Software, vol. 1, pp. 75-88, 1984.
- [7] B. W. Boehm; J. R. Brown; M. Lipow, "Quantitative evaluation of software quality, " ICSE '76 Proceedings of the 2nd international conference on Software engineering, pp. 592-605

- [8] Batool, A., Motla, Y.H., Hamid, B., Asghar, S., Riaz, M., Mukhtar, M., Ahmed, M.: Comparative study of traditional requirement engineering and Agile requirement engineering (2013)
- [9] Begel, A.; Nagappan, N., "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007., pp.255-264, 20-21 Sept. 2007 doi: 10.1109/ESEM.2007.12
- [10] Benediktsson, O.; Dalcher, D.; Thorbergsson, H., "Comparison of software development life cycles: a multiproject experiment," Software, IEE Proceedings - , vol.153, no.3, pp.87,101, June 2006
- [11] Bo Wei; Zhi Jin; Lin Liu, "A Formalism for Extending the NFR Framework to Support the Composition of the Goal Trees," Software Engineering Conference (APSEC), 2010 17th Asia Pacific , vol., no., pp.23,32, Nov. 30 2010-Dec. 3 2010
doi: 10.1109/APSEC.2010.13
- [12] Chapin, N., "Agile methods' contributions in software evolution," 20th IEEE International Conference on Software Maintenance, 2004. Proceedings. pp.522,, 11-14 Sept. 2004
doi: 10.1109/ICSM.2004.1357864
- [13] "Cysneiros, L.M.; Sampaio do Prado Leite, J.C., ""Nonfunctional requirements: from elicitation to conceptual models," IEEE Transactions on Software Engineering, vol.30, no.5, pp.328,350, May 2004

doi: 10.1109/TSE.2004.10

[14] Farid, W.M.; Mitropoulos, F.J., "NORMATIC: A visual tool for modeling Non-Functional Requirements in agile processes," Southeastcon, 2012 Proceedings of IEEE , pp.1,8, 15-18 March 2012

doi: 10.1109/SECon.2012.6196989"

[15] Farid, W.M.; Mitropoulos, F.J., "Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes," Southeastcon, 2012 Proceedings of IEEE , pp.1,7, 15-18 March 2012

doi: 10.1109/SECon.2012.6196988

[16] Feng Ji; Sedano, T., ""Comparing extreme programming and Waterfall project results," 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T) , pp.482,486, 22-24 May 2011

doi: 10.1109/CSEET.2011.5876129

[17] Gallardo-Valencia, R.E.; Olivera, V.; Sim, S.E., "Are Use Cases Beneficial for Developers Using Agile Requirements?," Fifth International Workshop on Comparative Evaluation in Requirements Engineering, 2007. CERE '07. pp.11-22, 16-16 Oct. 2007 doi: 10.1109/CERE.2007.2

[18] Gallardo-Valencia, R.E.; Sim, S.E., "Continuous and Collaborative Validation: A Field Study of Requirements Knowledge in Agile", Second International Workshop on Managing Requirements Knowledge (MARK), pp.65-74, 1-1 Sept. 2009 doi: 10.1109/MARK.2009.3

[19] Gu Hongying; Yang Cheng, "A customizable agile software Quality Assurance model," 2011 5th International Conference on New Trends in Information Science and Service Science (NISS), vol.2, no., pp.382,387, 24-26 Oct. 2011

[20] Hamed, A.M.M.; Abushama, H., "Popular agile approaches in software development: Review and analysis," 2013 International Conference on Computing, Electrical and Electronics Engineering (ICCEEE), pp.160, 166, 26-28 Aug. 2013

doi: 10.1109/ICCEEE.2013.6633925"

[21] Hashmi, S.I.; Jongmoon Baik, "Software Quality Assurance in XP and Spiral - A Comparative Study," ICCSA 2007. International Conference on Computational Science and its Applications, 2007. , vol., no., pp.367, 374, 26-29 Aug. 2007

doi: 10.1109/ICCSA.2007.65

[22] Hellmann, T.D.; Chokshi, A.; Abad, Z.S.H.; Pratte, S.; Maurer, F., "Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences," Agile Conference (AGILE), 2013 , pp.32,41, 5-9 Aug. 2013

doi: 10.1109/AGILE.2013.10

[23] Host, M.; Runeson, P., "Checklists for Software Engineering Case Study Research," First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007., pp.479-481, 20-21 Sept. 2007

doi: 10.1109/ESEM.2007.46

[24] Kassab, Mohamad, "An Empirical Study on the Requirements Engineering Practices for Agile Software Development," 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) , pp.254,261, 27-29 Aug. 2014

doi: 10.1109/SEAA.2014.77

[25] Lan Cao; Ramesh, B., "Agile Requirements Engineering Practices: An Empirical Study," Software, IEEE, vol.25, no.1, pp.60-67, Jan.-Feb. 2008 doi: 10.1109/MS.2008.1

[26] Layman, L.; Williams, L.; Cunningham, L., "Exploring extreme programming in context: an industrial case study," Agile Development Conference, 2004, pp.32,41, 22-26 June 2004

doi: 10.1109/ADEV.2004.15

[27] Lin Liu; Tong Li; Fei Peng, "Why Requirements Engineering Fails: A Survey Report from China," 2010 18th IEEE International Requirements Engineering Conference (RE), pp.317, 322, Sept. 27 2010-Oct. 1 2010 doi: 10.1109/RE.2010.45

[28] Lopez, C.; Cysneiros, L.M.; Astudillo, H., "NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge," Managing Requirements Knowledge, 2008. MARK '08. First International Workshop on , vol., no., pp.1,10, 8-8 Sept. 2008

doi: 10.1109/MARK.2008.7

[29] Macias, F "Empirical Assessment of Extreme Programming," PhD thesis, Department of Computer Science, University of Sheffield, 2004.

[30] Manjunath, K.N.; Jagadeesh, J.; Yogeesh, M., "Achieving quality product in a long term software product development in healthcare application using Lean and Agile principles:

Software engineering and software development," 2013 International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), , pp.26,34, 22-23 March 2013

doi: 10.1109/iMac4s.2013.6526379

[31] Martakis, A.; Daneva, M., "Handling requirements dependencies in agile projects: A focus group with agile software development practitioners," 2013 IEEE Seventh International Conference on Research Challenges in Information Science (RCIS), pp.1,11, 29-31 May 2013

doi: 10.1109/RCIS.2013.6577679

[32] Ming Huo; Verner, J.; Liming Zhu; Babar, M.A., "Software quality and agile methods," Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004., pp.520-525 vol.1, 28-30 Sept. 2004 doi:

10.1109/CMPSAC.2004.1342889

[33] Mitchell, S.M.; Seaman, C.B., "A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review," 3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009., vol., no., pp.511,515, 15-16 Oct. 2009

doi: 10.1109/ESEM.2009.531422

[34] Mnkandla, E.; Dwolatzky, B., "Defining Agile Software Quality Assurance," International Conference on Software Engineering Advances, pp.36,36, Oct. 2006 doi:

10.1109/ICSEA.2006.261292

- [35] Orr, K., "Agile requirements: opportunity or oxymoron?," *Software, IEEE* , vol.21, no.3, pp.71-73, May-June 2004 doi: 10.1109/MS.2004.1293075
- [36] P. Clarke and R.V. O'Connor, "The Situational Factors that Affect the Software Development Process: Towards a Comprehensive Reference Framework," *Information and Software Technology*, vol. 54, no. 5, 2012, pp. 433-447.
- [37] Paetsch, F.; Eberlein, A.; Maurer, F., "Requirements engineering and agile software development," *Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings*, pp.308-313, 9-11 June 2003 doi: 10.1109/ENABL.2003.1231428
- [38] R. Grau, K. Lauenroth, B. Bereza, E. van Veenendaal, and S. van der Zee, "Requirements engineering and agile development-collaborative, just enough, just in time, sustainable," 2014.
- [39] Racheva, Z.; Daneva, M.; Buglione, L., "Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products," *Second International Workshop on Software Product Management, 2008. IWSPM '08.*, pp.49,58, 9-9 Sept. 2008 doi: 10.1109/IWSPM.2008.7
- [40] Royce, Winston, W. "Managing the development of large software systems". Retrieved from <http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf>
- [41] Savolainen, J.; Kuusela, J.; Vilavaara, A., "Transition to Agile Development - Rediscovery of Important Requirements Engineering Practices," *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pp.289-294, Sept. 27 2010-Oct. 1 2010 doi: 10.1109/RE.2010.41

[42] Scharff, C., "Guiding global software development projects using Scrum and Agile with quality assurance," 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), pp.274,283, 22-24 May 2011

doi: 10.1109/CSEET.2011.5876097

[43] Singh, B.; Kannoja, S.P., "A Review on Software Quality Models," 2013 International Conference on Communication Systems and Network Technologies (CSNT), pp.801, 806 , 6-8 April 2013 doi: 10.1109/CSNT.2013.171

[44] Sommerville,I; Software Engineering (9th Edition) Harlow, England; New York Addison- Wesley, Mar 3 2010

[45] Soundararajan, S.; Arthur, J.D., "A Soft-Structured Agile Framework for Larger Scale Systems Development," 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2009. ECBS 2009., pp.187-195, 14-16 April 2009 doi: 10.1109/ECBS.2009.21

[46] STANDISH GROUP - 2013 The CHAOS Manifesto–Think Big, Act Small

[47] Sureshchandra, K.; Shrinivasavadhani, J., "Moving from Waterfall to Agile," Agile, 2008. AGILE '08. Conference , vol., no., pp.97,101, 4-8 Aug. 2008

doi: 10.1109/Agile.2008.49"

[48] Taehoon Um; Neunghoe Kim; Donghyun Lee; Hoh Peter In, "A Quality Attributes Evaluation Method for an Agile Approach," Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on , pp.460,461, 23-25 May 2011

doi: 10.1109/CNSI.2011.93

[49] Thayer, R.H., and M. Dorfman: Software Requirements Engineering. 2d ed., IEEE Computer Society Press (1997)

[50] Tsun Chow, Dac-Buu Cao, "A survey study of critical success factors in agile software projects," Journal of Systems and Software, v.81 n.6, p.961-971, June, 2008

doi>10.1016/j.jss.2007.08.020

[51] Vijayasathy, L.; Butler, C., ""Choice of Software Development Methodologies - Do Project, Team and Organizational Characteristics Matter?,"" Software, IEEE , vol.PP, no.99, pp.1,1

doi: 10.1109/MS.2015.26

[52] Waldmann, B., "There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development," Requirements Engineering Conference (RE), 2011 19th IEEE International , pp.301,305, Aug. 29 2011-Sept. 2 2011

doi: 10.1109/RE.2011.6051626

[53] Wieringa, R., "Towards a unified checklist for empirical research in software engineering: first proposal," 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), pp.161,165, 14-15 May 2012

doi: 10.1049/ic.2012.0020

[54] Yi Liu; Zhiyi Ma; Rui Qiu; Hongjie Chen; Weizhong Shao, "An Approach to Integrating Non-functional Requirements into UML Design Models Based on NFR-Specific Patterns," 2012 12th International Conference on Quality Software (QSIC), pp.132,135, 27-29 Aug. 2012

doi: 10.1109/QSIC.2012.23

[55] Yin, R. K. "Case study research: Design and Methods (3rd Edition)". Thousand Oaks, CA: Sage. 2003