# An Empirical Study from a Quality Perspective on the Impact of Switching from Waterfall to Agile

Anzira Rahman, Luiz Marcio Cysneiros
School of Information Technology
York University
Toronto, Canada
cysneiro@yorku.ca

***Abstract*— Agile Methods are characterized as flexible and readily 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 the use of these methods jeopardizes quality. 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 methods in 2012. This work used real life data from real life projects and focused on the registered number of defects for each project. Our results suggest that the adoption of Agile methods may negatively impact quality, in the long run. The results also suggest that Non-Functional Requirements may play an important role in this adverse effect.**

***Index Terms*—Agile Methods, Waterfall, Quality, Non-Functional Requirement.**

## I. INTRODUCTION

In information-intensive societies, 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 efficiency and faster turnaround are expected, software quality should not be compromised.

Many different software development methods have been introduced and used by the software engineering community since the late 1960's [25][29]. These methods have been improved and refined over time [25]. 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 them, and it is divided into linear stages [25]. In contrast, the Agile software development method is a more modern approach and comprised of a group of activities that encourages dynamic planning, development, and frequent delivery of artefacts to meet client satisfaction [3] [19] [14].

Since Agile software development methods focus on minimal documentation, Paetsch et al. [25] have suggested that Agile and Waterfall software method seem contradictory. In fact, they are similar with respect to the performance of primary activities, such as elicitations, management, and validations, but differ in terms of when and how these activities are executed. More specifically, waterfall is dependent on documentation for knowledge sharing, whereas Agile methods are less document-centric and prioritize face-to-face collaboration. In Agile methods, teamwork is conducted within a short timeframe to gather requirements, develop, test, and deliver [19]. 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 [30] [24]. Thus, one well-recognized benefit of Agile 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 [31]. Though, in Agile methods, validations happen throughout the process to ensure quality [14] [25], the easier accommodation of changes in Agile is typically completed without sufficient documentation, which may lead to challenges in keeping up with the evolution of requirements while maintaining software quality.

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 [31]. To date, there has not been sufficient study regarding the actual practices of software professionals for software activities using Agile methods [18]. Also, empirical comparisons between different software development processes have been minimal [26]. Perhaps most importantly, most of the works are based on the perception of software developers instead of real data. In this work, we used data extracted from logs of defects collected from several software development projects. It compared the impact of software development methods on quality using software development projects carried out 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

methods in 2012. The log of defects reflects all the defects that were found and corrected after the software was deployed.

The goal of this study is not to advocate one method or another. The primary objective is to use data extracted from several real life projects to contribute to understanding if the adoption of Agile methods impact quality negatively or not. By extension, this study can contribute to other company's decision-making process when choosing what methods to adopt and apply. Although our results reflect only Agile and Waterfall methods, they can be used to infer possible consequences on other methods such as spiral or rational unified process. 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. Given that the log of defects contained explanations describing each defect, we also used the study to test if there was a significant difference in the number of defects due to missed or wrongfully specified Functional Requirements and Non-Functional Requirements (NFR) when comparing defects in Agile projects and defects in Waterfall projects,

The rest of this work is structured as follows. Section II will introduce some background positioning the reader on the subject. Section III will present the research method used in this work. Section IV will present our results, which will be discussed in Section V. Finally, Section VI will conclude the work and point out to future work.

## II. Background

Though the purpose of Requirements Engineering (RE) is the same in all software methods, how RE is performed in Agile and Waterfall methods is opposing in nature [3] [24]. RE in both Agile and Waterfall methods place importance upon stakeholder involvement, face-to-face collaboration, and delivering high-quality products [14] [24]. However, while customer collaboration and communication are crucial for achieving a good quality product [4] [6] [14] [19], 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 [24].

Agile and Waterfall methods are also different with respect to their use of Non-Functional Requirements. In Agile approaches, NFRs are not properly identified, modeled or linked during the early requirement analysis phase [11] [12]. Due to the nature of evolving requirements, NFRs such as maintainability, portability, safety or performance are often overlooked in Agile [25]. In addition, NFRs are not often seen as crucial or highly critical by teams using Agile methods [26] [10].

Perhaps most importantly, Agile and Waterfall methods differ in terms of their use of documentation. Agile methods are more code-oriented [26] than Waterfall method and therefore requirement knowledge is not captured in comprehensive documentation. According to Chapin [9], the vast majority of Agile methods accomplish software maintenance without documentation. User stories do not become part of the complete documentation of the system [19] [14] 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 [14], and documentation is used to share knowledge [3] [24].

### *A. 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 [15] regardless of the software development method employed [16]. Sommerville [29] 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, and portability among others [29]. It also refers to the process of evaluating software to ensure compliance with software requirements [3] [28]. SQA activities include auditing, testing of the software and an awareness checkpoint to examine the project's status [27]. According to Bohem [5], 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 the goods in all industries [2].

The use of SQA in Agile and Waterfall methods is different in terms of the initiation, frequency and focus of the activities. In Agile, SQA activities start from the very beginning whereas, in Waterfall, SQA activities typically do not begin until all developments are completed. In the field study conducted by Manjunath, Jagadeesh, and Yogeesh [23], the ability to detect issues earlier was noted to provide chances to fix them at a previous stage.

A second difference between Agile and Waterfall methods lies in how frequently the SQA activities are performed. According to Cao and Ramesh [17], early and constant validation and frequent review meetings are used to validate requirements in Agile. Also, Mnkandla and Dwolatzky [26] concluded that quality assurance occurs more frequently in agile methods 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 [16].

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 usually concentrates on testing the full requirements of the project. In Agile, unit testing and acceptance testing are the primary focus [17]. In general, there seems to be a lack of significance put in for Non-functional requirement [10] in Agile than Waterfall. According to

Taehoon et al., previous research with Agile approaches did not focus on non-functional aspects and suggested that identifying non-functional requirements early in the project can contribute to achieving improved quality [32].

Aspects related to issues such as the length of the project and cost could be considered when one is evaluating the quality of software. However, since those are attributes that are rated very differently from one company to another a statistical treatment for this would not be effective and therefore we focused on the number of defects as our primary item for comparison.

## III. Research Method

Qualitative research methods are useful for exploring and understanding any emerging research problem. 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 [14]. Many of the qualitative studies that were completed helped to gain an understanding of the complex underlying reasons, opinions, and motivations for using one method over another [19]. They investigated the why rather than the how many [3]. In contrast, a quantitative approach can be used to quantify the problem by way of generating numerical data to transform into useable statistics and to evaluate and analyze the findings.

While reviewing the literature, it is evident that there are many studies on understanding 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 adopting Agile methods on software quality, especially comparing the number of defects as part of the quality assessment.

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.

In this work, a multiple-case (completed for multiple organizations) with an exploratory method was selected. With respect to the selection of an organization, a company specialized in transaction management for mortgage processing was chosen as the identified case. The primary reason for selecting this 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 as well as the fact that one of the authors works there. 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.

The aim of this case study was to determine whether there is a difference in the number of defects 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 in Agile and Waterfall projects is different.

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

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

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

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

### *A. 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 important to understanding which data was to be collected, from where and why it was required for testing the hypotheses.

#### *1) 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 department known to provide technological solutions to a diverse range of clients over a number of years under Waterfall and Agile methods. This particular department 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.

#### *2) 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 department 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 allocated to the Waterfall group. The year of

initiation of a project was used to assign a project to each group. All projects were double-checked to assure they were correctly classified.

*3) 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 kept a detailed history of all projects including the number of requests (user stories and work items). It also kept track of the number of defects detected during the software life cycle.

For the purpose of this work, the data from TFS was accessed and collected using several steps. First, the number of requests 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 the description of the defects were also collected. Table 1 shows a sample report of defects from TFS.

Table 1: Sample report of defects in a project

| ID | Type | Hours Spent | Title | Found During |
|---|---|---|---|---|
| 676 | Defect | 7.5 | Admin approved users are getting managerUserIDs overridden by CurrentSystemUserID | Dev/ Functional Testing |
| 683 | Defect | 5 | Portal Self Registration for Sales Force and Mortgage Closing Officer- Should not be able to register without a Branch Manager | Stage/ Internal Acceptance Testing |
| 616 | Defect | 3 | Self Registration "Manage Pending User" link having translation issue | Prod/ Deployment |

The selected projects allowed comparisons to be made between Agile and Waterfall methods to understand the overall impact of each method in terms of quality.

*4) Criteria for Case Selection:*

In order to select the cases, the characteristics of the project and teams were important. According to the survey done by Vijayasarathy and Butler [33], 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. In the end, our sample was composed of 8 Agile projects and 8 Waterfall projects.

The criteria used for selection were: project size, team size, and duration of the project.

*a) Project Size:*

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. An explanation of what user stories and work items mean in this organization is presented below:

**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 [3] [23]. A user story is structured in concise format that describes in natural language the "Why" and "How" of a project [13]. The user stories are typically written in the form of "As a (Role), I want (Something) so that I can (Benefit)". 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.

**Work Items:**

In projects completed under the Waterfall method, a traditional format was presented that included a Business Requirement Specification Document (BRSD). The BRSD is a written plan completed before beginning the 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 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 1 and Figure 2).

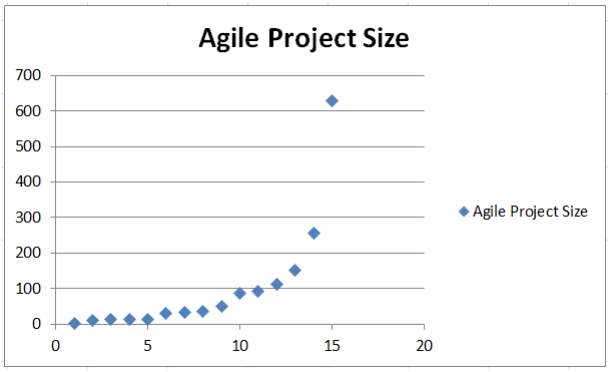


Fig. 1: Size of selected projects using an Agile method

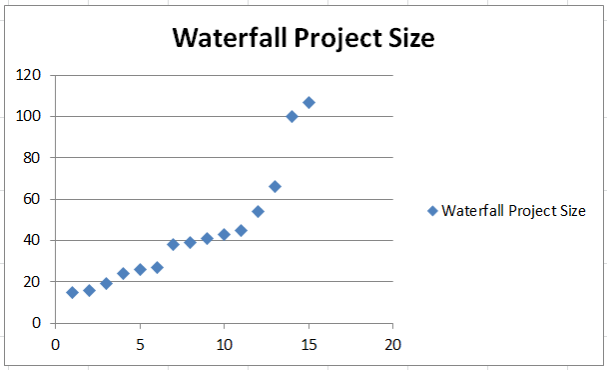


Fig. 2: Size of selected projects using an Agile method

*b) Team Size:*

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 department consisted of 35-40 members throughout the years that the projects used in this study were developed. This number was 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 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. This was also an attempt to minimize bias towards either Agile or Waterfall.

*c) Duration:*

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. Possible exclusions were based on how the department sees small, medium and long projects. Within the fifteen randomly selected projects in the Agile group and the fifteen randomly chosen in the Waterfall group, duration ranged from one month to two years. Projects that had duration of three-nine months (considered medium projects) were found to be suitable for inclusion, and cases with a smaller duration (one-two months) and longer durations (ten-twenty four months) were excluded.

*B. Dependable Variables:*

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. 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.

*1) Total 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.

*2) Number of NFR related defects:*

Defects that after analyzed were considered caused by a missing or wrongfully defined Non-Functional Requirement.

*3) Number of FR Related defects:*

Defects that after analyzed were considered caused by a missing or wrongfully defined Functional Requirement.

*C. 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. In order to compare the difference between the means, a t-test was performed on two independent groups (Agile and Waterfall) for each dependable variable. While exploratory studies are useful to examine novel trends in small group, a conservative alpha level can introduce bias toward type II error (failure to detect a significant finding) and so effects may be hard to detect in small samples. Given the small sample size (n= 8 per group), a more liberal alpha level was chosen ($\alpha$ =0.10). All statistical analyzes were completed using IBM SPSS Statistics Version 22.

## IV. RESULTS

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. Quality comparison from the perspective of duration of the project and budget implications was left to be carried out on future work.

### *A. 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 3 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.

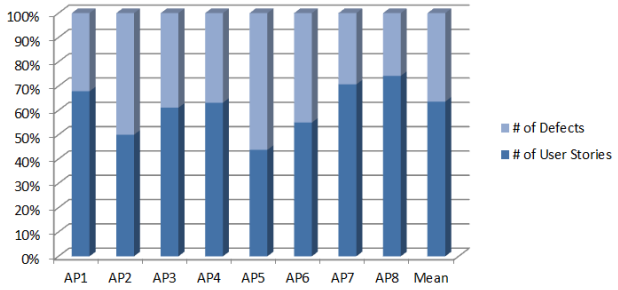


Fig. 3: Agile Projects: Proportion of User Stories and Defects

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 4 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.

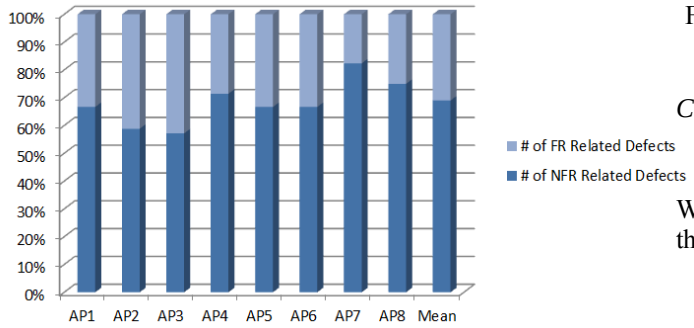


Fig. 4: Agile Projects: Defects with proportion of NFR and FR related defects

### *B. 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 5 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.

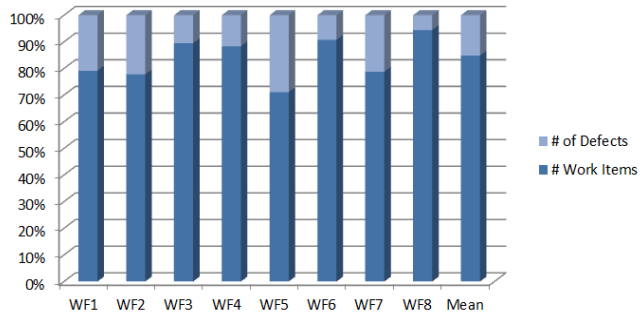


Fig. 5: Waterfall projects: Proportion of work items and defects

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 6 illustrate the proportion of NFR and FR defects in each Waterfall project.

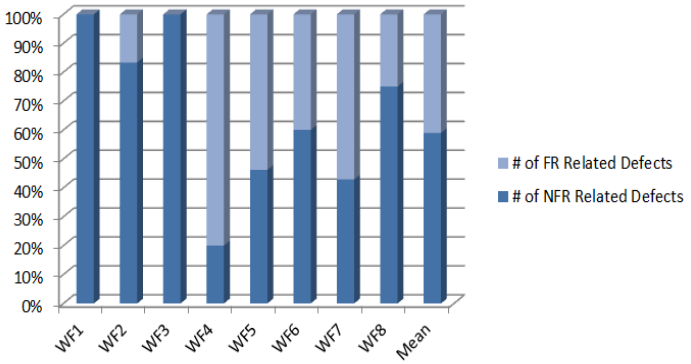


Fig. 6: Waterfall projects: Defects proportion of NFR and FR related defects

### *C. 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.

#### *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); t (14) =1.971, p = 0.069. Projects using Agile method had significantly more defects than projects using Waterfall method. The alternative hypothesis was accepted, in this case.

*2) Hypothesis 2:*

There was a significant difference in the number of defects for Agile (M=8.6, SD=3.8) and Waterfall (M=4.1, SD=1.7); t (14) = 3.039, p =0.009. Agile had more NFR-related defects than Waterfall method that suggested that the non-functional requirements are harder to be correctly elicited ad dealt with in Agile methods. Hence, the alternative hypothesis was selected.

### *D. Strengths of the Research*

Although there are general studies on Requirement Engineering and Software Quality Assurance regarding Agile methods, 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.

One of the strengths of this work lies in its examination of materials in the real world (i.e., ecological validity). This has practical applications that are relevant and useful to be applied by companies. This type of research is hard to duplicate in a controlled environment with limited resources.

Secondly, this study accessed a combination of projects that catered to different clients in the financial industry. While we cannot be certain our results can be extrapolated to other sectors we believe that many other domains share enough similarities with the financial area to use our findings as guideline for future decisions. Also, having different clients for each project offers feedback from multiple sources.

Lastly, we had the rare opportunity to extract data that could allow us to analyze the defects and to differentiated the type of defects between those related to Functional Requirements and those due to Non-Functional Requirements. Despite the lack of works in how to deal with NFR during Agile projects, companies adopting Agile methods may be able to include NFR elicitation and modeling in an ad-hoc manner. They may also try to adapt existing proposal to deal with NFR in non-agile methods to their Agile processes. 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.

### *E. Limitations of the Research*

The aim of this work 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.

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.

We recognize that the sample size is not the ideal, but a scenario as the one that we encountered to extract our data is rare and should not be dismissed due to a non-optimal sample size. Although we could have used fifteen projects of each type instead of eight, we understand that the gain in sample size would not compensate the benefits of obtaining a sample that would be less prone to bias and more representative of a general case scenario that we might face in most companies.

## V. DISCUSSION

We found one significant prior study showing that Waterfall method provided better quality than Agile method. It was performed by The Standish Group, which has been collecting case information on real-life IT environments and software development projects since 1985. They have examined project success and failure in more than 90,000 IT projects and 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 with less than the required features and functions [31].

On the other hand, some studies have suggested that Agile methods have advantages over Waterfall method. For example, in an analysis, Ming et al. [25] 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 [25].

Another case study [20] that compared two releases of the same product completed by the same personnel, revealed a 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 work. Significantly smaller release may have contributed towards quality improvement of XP over Waterfall. However, our findings suggest that when using a more inclusive and less biased sample, Agile may 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 [7]. Macias 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 [22].

Our results suggest that, in fact, the adoption of Agile methods may lead to higher number of defects and in consequence, to higher cost to maintain the software that could offset the gains of productivity during development. While our study cannot be used to explain why the number of defects was greater in Agile it was not our goal to do so. We believe that the qualitative works mentioned before have already discussed the possibilities. Taking into consideration that many defects in Agile projects may not have been registered in TFS due to the minimum documentation principle, the impact of adopting Agile in quality might be even higher than what we have measured. Considering also that fixing a problem later, rather than sooner, in the software development life cycle can be two hundred times more expensive, it seems probable that a future work that can measure all these facts together may conclude that the final costs of projects adopting Agile may be higher that Waterfall. It is also important to register that we compared Agile with Waterfall a relatively rigid approach. It would be interesting to see results if we can replicate this experiment in an environment where Agile can be compared with other methods such as Spiral or Rational Unified Process. It is important to emphasize that during the selection of projects to be included in our samples we focused on medium-size-complexity projects. Small projects might produce a different result, but we did not have enough samples to run any statistical work tackling this scenario.

Some works support that Agile methods deal less with NFRs when compared to the Waterfall method [11][12][32]. 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 [21]. 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 [32]. Also, Agile have not adequately modeled Non-Functional Requirements (NFRs) and their potential solutions (operationalization) in early development phases [11]. 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 be used to 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 [18] that 68% of Agile projects reported the use of NFR during estimation compared to 40% for Waterfall. These data suggest that Agile methods involve more NFRs than might be expected from current understanding when compared to Waterfall method. However, our results, 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 [18], 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 Agile method do not meet the need for carefully eliciting and modeling NFR. b) NFR operationalization typically involves negotiations among several stakeholders and compromises to be made in requirements. The fast pace of Agile development and the minimal documentation principle may prevent developers to address all the possibilities to implement comprehensive and equitable solutions to satisfice NFR.

The findings of this work support the need for continued attention to understanding, documenting and communicating requirements in the software development process. As mentioned before, 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 [24] and doubt the benefits [16]. Although Paetsch [25] 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.

One day before concluding this paper, we were notified that the company from where the data was extracted is considering keeping Agile development only for small projects and return to waterfall for medium and large projects.

## VI. CONCLUSION

This study compared the impact of switching from Waterfall development process to an Agile one. It used a company where the software development department 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 that were worked on by medium-sized teams, and spanned between three and nine months in length were identified as project for inclusion. We used the number of defects as the variable to define quality in these projects. An application lifecycle management system, called Team Foundation Server (TFS), was used to track and store the information for this company such as the number of requests and the number of defects.

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 significantly more NFR-related defects than Waterfall. The expressive larger number of defects in Agile point out the possibility that gains due to a faster and leaner development process may be offset later due to the time and effort necessary to fix resulting defects.

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. A framework to deal with NFR in Agile Methods will also be one of our priorities.

## References


[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] 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

[3] 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

[4] B. Boehm, "Requirements That Handle Ikiwisi, COTS,and Rapid Change," Computer, July 2000, pp. 99–102.

[5] B. W. Boehm, "Verifying and Validating Software Requirements and Design Specifications," IEEE Software, vol. 1, pp. 75-88, 1984.

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] Farid, W.M.; Mitropoulos, F.J., "NORMATIC: A visual tool for modeling Non-Functional Requirements in agile processes,", 2012 Proceedings of IEEE Southeastcon, pp.1,8, 15-18 March 2012 doi: 10.1109/SECon.2012.6196989"

[12] Farid, W.M.; Mitropoulos, F.J., "Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes," 2012 Proceedings of IEEE Southeastcon, pp.1,7, 15-18 March 2012 doi: 10.1109/SECon.2012.6196988

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

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

[20] 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/ADEVC.2004.15

[21] 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

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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] Singh, B.; Kannojia, 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

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

[30] 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

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

[32] 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

[33] Vijayasarathy, L.; Butler, C., ""Choice of Software Development Methodologies - Do Project, Team and Organizational Characteristics Matter?,"" IEEE Software, vol.PP, no.99, pp.1,1 doi: 10.1109/MS.2015.26

[34] 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