DETERMINING THE INFLUENTIAL FACTORS OF CONDUCTING NON-FUNCTIONAL TESTING IN AGILE SOFTWARE DEVELOPMENT

ASHWINESH C GANAPATHI

A Dissertation Submitted to the College of Graduate Studies, Universiti Tenaga Nasional in Fulfilment of the Requirement for the Degree of

Master of Software Engineering

MARCH 2020

DECLARATION

I hereby declare that the dissertation is my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously, and is not concurrently submitted for any other degree at Universiti Tenaga Nasional or at any other institutions. This thesis may be made available within the university library and may be photocopied and loaned to other libraries for the purpose of consultation.

_____

ASHWINESH C GANAPATHI

Date : 6TH MARCH 2020

ABSTRACT

Non-functional requirements define the systems attributes such as performance, security, etc. Even though these requirements are important in a system, they are known to be left out due to various factors, most commonly due to time and budget of a project. Agile software development practices have become a preferred approach in software development due to the ability to deliver an end product at a short period of time. However, in agile software development, there is no phase specifically defined for requirement elicitation which diverts the agile team members focus away from non-functional requirement testing and focus is put mainly toward functional testing instead. In this study, we identify the factors influencing non-functional testing, we discover the challenges faced in conducting non-functional testing and what are the practices that can be adopted to successfully conduct non-functional testing agile software development projects. Therefore, a detailed literature review was conducted to identify the factors, challenges and practices gathered from previous studies to be included in the survey. A quantitative approach was used whereby a set of questionnaires consisting statements related to factors, challenges and practices were distributed in Malaysia. An expert review was conducted to validate the survey. SPSS Version 26 was used to analyze the data obtained from the survey whereby various statistical tests were administered. As a result, this study identified thirteen factors influencing non-functional testing, six main challenges faced in conducting non-functional testing and seven practices that can aid in the process of conducting non-functional testing in an agile environment. The findings from this study would benefit agile team members to have a better understanding of the significance of conducting non-functional testing and at the same time serve as a guide for agile team members in conducting non-functional testing.

ACKNOWLEDGEMENTS

Firstly, I would like to thank all the staff members of Universiti Tenaga Nasional for assisting me and providing me with the required knowledge during my master studies. I owe my deepest gratitude to my research advisor, Assoc. Prof. Dr. Roslan Ismail, whom guided and provided me with continuous support throughout the whole process of the dissertation.

Most importantly, I would like to thank my parents, C Ganapathi & Shamini, for their love and being the main pillars of support throughout my journey. Special thanks to Yoganesh, Dharshini, Dhaarini, P.A.B & Thashini for believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ASD	Agile Software Development
EFA	Exploratory Factor Analysis
FR	Functional Requirements
KMO	Kaiser–Meyer–Olkin
NFR	Non-Functional Requirements
RE	Requirements Engineering
SPSS	Statistical Package for the Social Sciences

CHAPTER 1

INTRODUCTION

1.1 Introduction

In this chapter, we explore the background of the study. Other than that, we dive into the problem statement and present the research goals, research questions, explain the importance of this study and finally the research scope. We listed down the structure of the paper at the end of this chapter.

1.2 Background of the Study

Agile development methodology has been taken notice and become a preferred approach to adopt by projects due to the ability of delivering high-quality software in a short timespan (Alexander T, 2018). The ability to lessen the developement period has made the process more preferred compared to the traditional software processes. Due to business processes evolving everyday as well as the complexity, IT firms strive to stay ahead of the competition by squeezing deadlines (Bose S, Kurhekar M and Joydip G, 2014). Teams adopting the agile methodology have the luxury of flexibility and changeability in their projects. The main characteristics of agile development are adaptability, people-oriented, more code-based and lesser documentation compared to traditional processes which are more predictive and process-based (Agile Manifesto, 2014). However, with all these benefits come risk, as the reduce in time period, results to expediting or leaving out certain processes; in this case, non-functional testing.

Software testing is a crucial process in a software development life cycle as it points out the failures or defects that were made during the development phase. It is the process of validating the system components or system requirements by using either manual methods or automation tools to confirm whether the requirements are fulfilled and to ensure the actual results are in line with the expected results (Itti H and Rajender S, 2015). While some errors are not damaging, others can lead to extremely expensive additional cost to the project. Software testing process involves creating test cases from the identified requirements to achieve the desired quality of the system, which are done by requirements and testing specialists. But in agile development, this cannot be done as agile development does not contain a phase to gather requirements and perform analysis. Therefore, no test cases will be created without a set of requirements to do so and without any test cases, software testing cannot be conducted.

Non-functional requirements define the systems attributes which includes security, performance, etc. As a matter of fact, the system's quality as a whole is defined by non-functional requirements. Identifying and prioritizing the non-functional requirements are known to be complex as noticed in the traditional software development methodology (Eliane, F. C, 2012) (Mylopoulos J, Chung L and Nixon B, 1992) and this is purely because the nature of non-functional requirements does not appear visually and does not have an effect on the system's features or functions. Technical requirements are what non-functional requirements are known as, and these technical requirements often relate and involve certain functional requirements (Ambler S. W, 2008). Without a doubt non-functional testing is significant in determining a project's success (Lawrence B, Wiegers K & Ebert C, 2001), however functional requirements are usually the priority in a project when compared to non-functional requirements (Martens N, 2011). Since, there's not a phase defined specifically for requirement elicitation in Agile Software Development (ASD), non-functional testing is known to be left out or considered as low risk.

A compilation of factors listed by various other researches are stated in the upcoming chapters.

1.3 Problem Statement

Agile development has changed the way of software development process. The ability to deliver a working software with a quick turnaround time span has always been the upmost priority in the customers perspective. Up to that point, requirements elicitation has been a key process in a software development process. It is the phase where a set of requirements of the system is collected for software testers to create test cases from the requirements to ensure the quality set is achieved. However, due to the structure of ASD, this is no longer feasible as there is no phase established for requirements elicitation to produce the requirements in order to proceed with the software testing process.

Although software testing will be conducted in the software development process, due to the fact that there is no proper set of requirements, it is clear that the software testing would not be conducted as thorough as needed. Software testing is a process of error identifying and the validation of the system's actual results with the expected results. One of the key processes of requirements engineering is discovering requirements in a software engineering design process (Hanan, H., Rocky, S., Jianwei, N., & Travis, D, 2017). Priority is always given to functional testing as it tests the behavior and execution of the system itself. In other words, the software should be able to do what it is intended to. Meanwhile, non-functional testing checks the system's ability to complete a certain task. This might seem like a background process but it is a critical aspect of a system as it concerns the performance and security of the system.

Non-functional requirements testing is not being considered as an important part of a software development process (Bahiya M and Abdelhamid M, 2015). They are placed into a low-risk category because of their characteristics as compared to functional requirements (R. Cristina, M. Sabrina and S. Daniela, 2016). Not conducting non-functional testing has become a norm till requested to do otherwise. When an issue

regarding the performance of the system is brought up by a customer, usually that is when the performance of the system will be looked into. Customers are the main priority when it comes to agile development. However, due to their limitation and knowledge of the background processes of a system, their focus would be more towards the business side and assume the technical aspects will be handled by the development team.

One of the common factors identified in the negligence of conducting non-functional testing is time (R. Cristina, M. Sabrina and S. Daniela, 2016). The need for more time to perform non-functional testing is the main challenge in ASD as the process focuses on rapid implementation and delivery which makes team member to ensure the functionality of the system works to be considered as a working system. By not performing non-functional testing, the overall quality of the software is at risk.

1.4 Research Questions

1) What are the reasons for agile team members to ignore or not conduct the testing of non-functional requirements?

2) What are the challenges faced by agile team member when conducting non-functional testing?

3) What are the practices used by agile team members to overcome obstacles when conducting non-functional testing?

1.5 Research Objectives

The overall objective of the study is to help agile team members to have a better understanding on the significance of non-functional testing. Listed below are specific objectives of the study:

1) To identify the influencing factors and challenges of conducting non-functional testing in agile software development.

2) To validate the factors influencing non-functional testing in agile software development.

3) To propose the practices that can be adopted by agile team member when conducting non-functional testing.

1.6 Significance of the Study

This research is important to help agile team members to have a better understanding on the significance of non-functional testing. We aim to determine the influencing factors of conducting Non-Functional Requirements (NFR) in ASD. By doing so, agile team members are able to plan ahead and expect the obstacles they might face in the process beforehand of conducting non-functional testing. This study shares the practices used by agile team members to ensure proper non-functional testing can be conducted. This study will help improve the understanding and knowledge of agile team members on the non-functional testing process.

1.7 Scope of the Research

Agile methodology is practiced in large number of projects due to its advantages but an important process that can help improve the end product is being overlooked which is the non-functional testing. In this study, we determine the influencing factors of conducting non-functional testing in ASD, what are the obstacles that they go through when conducting the process and what practices can be used to be able to conduct non-functional testing in their projects. Existing factors identified from previous studies were reviewed to be validated in this study as well. An expert review was conducted to validate the survey built to ensure that the data to be collected aligns with the objective of the study. Finally, Statistical Packages for the Social Sciences (SPSS) was used to analyzed the results gathered by using various statistical tests.

1.8 Structure of the Remaining Chapters

The remaining dissertation is organized as follows.

Table 1. 1 Structure of the Remaining Chapters

Chapter	Summary
2: Literature Review	In this section, we will dive into previous work that pertains ASD, software testing and non-functional testing. Mainly, we will review the factors influencing non-functional testing from previous studies. Other than that, we will include the challenges faced in conducting non-functional testing and the practices to overcome those challenges that were identified by several previous studies.
3: Methodology	This section focuses on the methodology utilized to conduct this study. It explains the direction in how the research plans to gather be able to answer the questions opposed in the research study. Other than that, data collection and data analysis methods will be stated.
4: Results and Analysis	In this section, we present the findings from the data collected. Statistical tests in SPSSS are used to analyze and validate the obtained data.
5: Discussion and Conclusion	In this section, we present the final findings attained from the analyzed data. This chapter also highlights implications, limitations and recommendations for the future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

We present the literature related to this study and mainly to the research objectives. Particularly, we highlight the definitions related to this topic. Furthermore, we will review the existing findings of factors influencing NFR testing.

2.2 Agile Software Development

ASD has impacted the way on how software is developed worldwide (D. Tore & D. Torgeir, 2009). In the IT industry, this is solely due to the fact that by utilizing the agile development method, companies are able to squeeze in tight deadlines for projects (S. Bose, M. Kurhekar and G. Joydip, 2014). The challenge software engineering faces these days is to produce a fully functioning system immediately upon request due to the increase in competition.

Comparing both agile and traditional methods, the traditional method abides a sequence where the project is completed by stages which means the project will not be able to move on without completing a current stage. Whereas, the agile methodology practices an iterative approach to rapidly deliver a complete system and during the process a particular stage can be revisited if required (D. Morelos, 2011).

There are many other differences between the two approaches and Table 2.1 highlights them (S. Nerur and V. Balijepally, 2007).

Table 2. 2 Difference between Traditional and Agile perspective on Software Development

	Traditional view	Agile perspective
Design process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving process	Selection of the best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/generative
Key characteristics	Control and direction Avoids conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication; integrates different worldviews Embraces conflict and dialectics Encourages exploration and creativity; opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical and/or philosophical roots	Logical positivism, scientific method	Action learning, John Dewey's pragmatism, phenomenology

Agile method allows customer to be engaged with throughout the development process where their inputs and suggestions for improvements will be gathered to ensure satisfactory in the finished product. As opposed to the traditional method where the involvement of the customers will take place at the initial phase but have limited involvement as soon as the software development process begins.

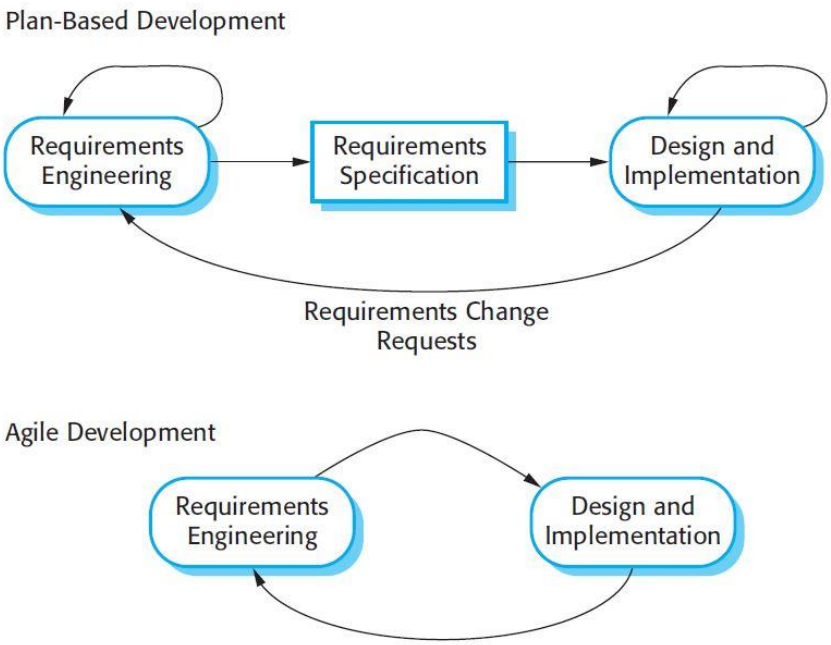

Figure 2. 1 Plan-Based Development and Agile Development (I. Sommerville, 2011)

Based on Figure 2.1, author Ian Sommerville explains that in an agile approach, the design and implementation activity is the heart of the project. Phases such as requirement elicitation and testing are integrated together with the design phase. Whereas traditional methods known as plan-driven depend on the output phase by phase to proceed and plan for following activities. In a nutshell, ASD does not have an established phase dedicated for requirement elicitation that can produce a set of requirements needed for testing process.

2.3 Agile Software Testing

Software testing examines whether the actual output matches the expected output after executing a software system and ensuring the system is defect free (S. Karuturi and G. Malle, 2017). Software testing is significant in a project to identify bugs or errors in the system. Once an error determined is fixed, the system needs to be tested again to confirm the fix. Half of the development cost will be required to cover the software testing process, which identifies it as the phase which cost the most amount in a software development (Kit. E and Susannah. F, 1995). While large sums of money are invested in software testing, if the process is done correctly, the effectiveness may not be as much as one would consider, mostly 20 percent (Huang. L and Boehm. B, 2006).

Traditional testing has been a standard method in performing software testing. The testing is done in a set of phases where completion of one phase is required to proceed to the next phase. Whereas in agile testing, the process follows an iterative approach and an adaption model. Table 2.2 states how these two differ.

Table 2. 3 Difference between Traditional and Agile Software Testing

Traditional Testing	Agile Testing
Testing is executed in phases using a top-down approach	Testing is executed in an iterative approach and an adaptive model
Testing is done only after the development process is completed	Testing is done on-the-go where the defects are fixed at each sprint
The requirements stated are finalized and not easily changed	The requirements are fixed but flexible to adapt to changing business and user requirements
Changes and modifications are usually implemented on the next release	Changes and modifications can be implemented on the next sprint
Feedbacks are taken from the end users once the product is completed	Feedbacks are taken from time to time when process is ongoing to ensure client satisfaction
Time consuming as one whole phase is dedicated for testing	Prevents spending excessive time as testing is done at each sprint

Both testing approach can be effective and efficient in their own way. Comparing the approaches might make it seem that one has an advantage on the other, both will ensure results. The choice of implementation solely depends on the requirements of the project as well as the client.

However, the focus will be on agile environment for this paper. The testing process in an agile environment has been gathering attention as agile practices depend on accurate software testing since the very beginning (Glenn V, 2005). Agile operates on incremental method. In this case, testing has to be executed on at each stage during development (Rijwan K, Akhilesh K and Dilkeshwar P, 2016). Due to the fact that agile processes are iterative, test activities are required to be executed quick and efficient following the iterations (Collins E and Lucena V, 2010). Agile testing means testing an application with the mindset of understanding it and allowing customers to be involved in guiding the testing in line with agile principles (Crispin L and Gregory J, 2009). According to these authors, these are the important aspects required to perform agile testing successfully as shown in Figure 2.2.

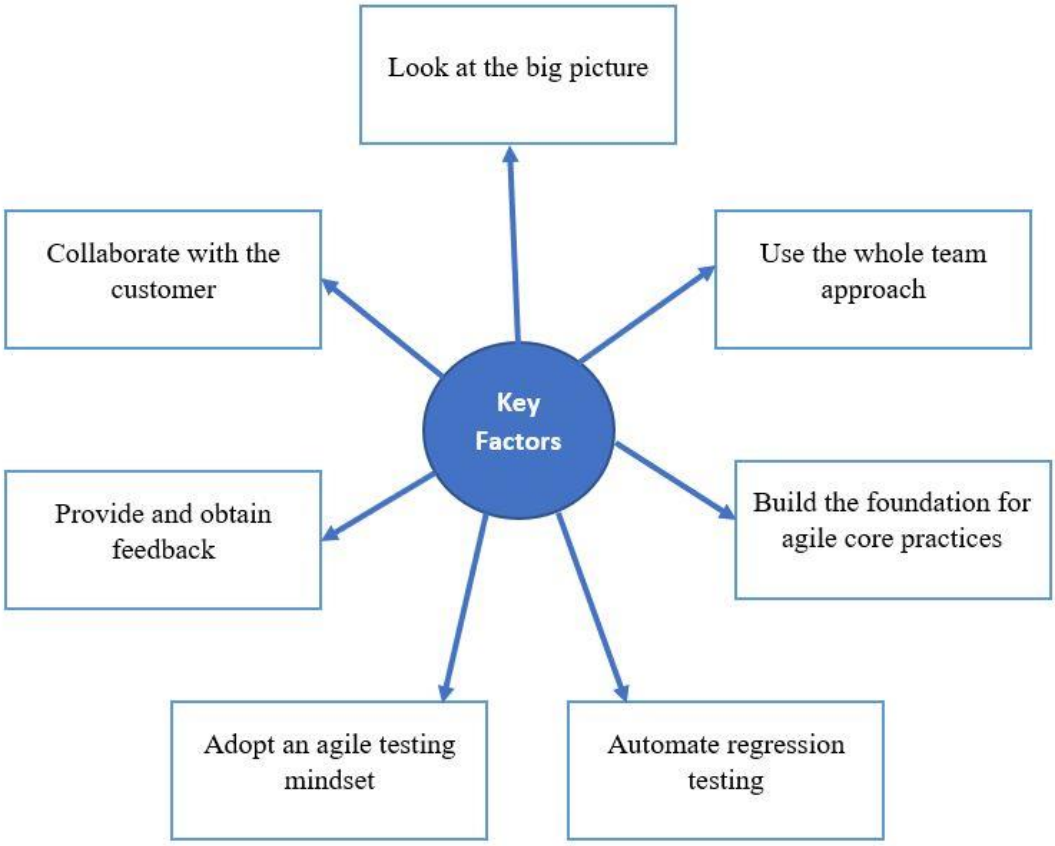

Figure 2. 2 Key Factors to Perform Agile Software Testing

Customers involvement is the focal part of ASD. But generally, they might lack knowledge on the technical aspect of the NFR and concentrate more towards the business side of the system (R. Cristina, M. Sabrina and S. Daniela, 2016). Based on Crispin and Gregory, business partners trust that the development team should handle the NFR aspects such as performance and security of the system. Figure 2.3 presents the various types of agile testing viewed from different angles (Crispin L and Gregory J, 2009).

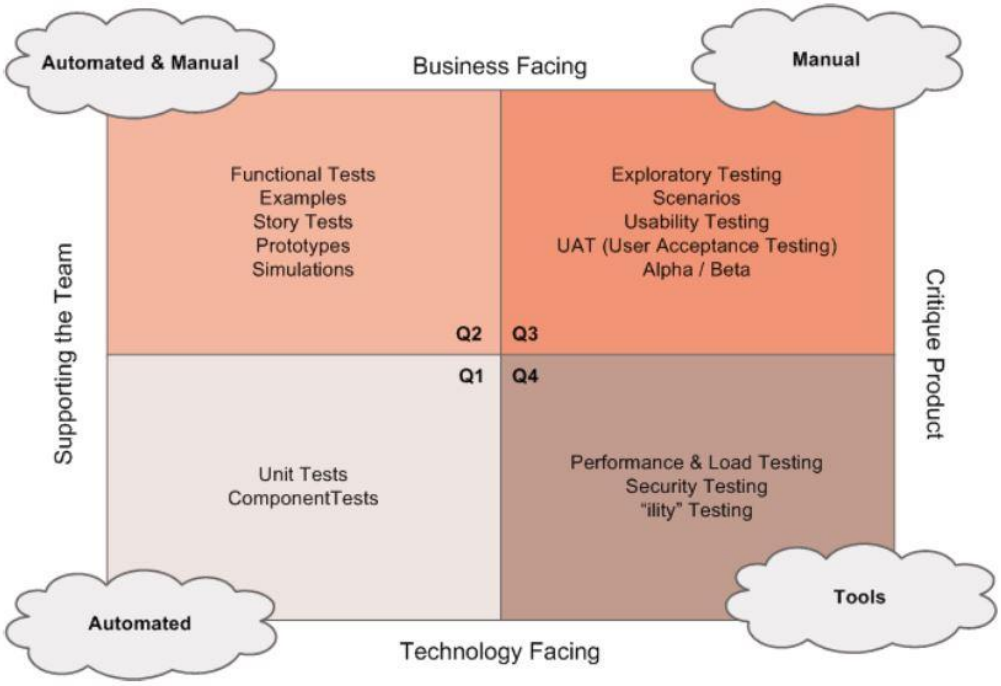

Figure 2. 3 Agile Testing Quadrants (Crispin L and Gregory J, 2009)

The first and second Quadrant tests lean towards specifying the requirements and the designing process. Quadrant one's tests which consists of unit and component tests are executed in a regular basis to ensure the quality of codes. Tests in Quadrant 2 involves requirements design. Examples, simulations, story tests and prototypes are executed to verify whether the purpose of the application was as desired. Whereas the functional tests ensure the codes functions as required.

Tests conducted in the third and fourth Quadrant judges the application in a different point of view. Testing in the third Quadrant involves business scenarios. It is conducted to check if the functionality of the application meets the required business demand. The tests in Quadrant 4 are more technical related which requires accurate analysis of results and usage of technical tools to assist in the process. These tests in Quadrant 4 requires specialized knowledge to be conducted correctly.

Although there are many types of software testing, this paper will only focus on the testing of NFR.

2.4 Non-Functional Requirement

Requirements engineering (RE) is a process of identifying what is required by a system, followed by documentation of the process and maintaining those gathered requirements (Raimundas, 2005) (Pamela, 1997) in a software engineering design process. System requirements describes the features and behavior of a software application. Traditionally, the procedures of RE is a set of sequential activity. The initial stage of the process is *requirements elicitation*, where details on the requirements are gathered. After the information is gathered, the *requirements analysis* can begin, where an understanding is created on the requirements. The following stage is *requirements specification*, and this is the phase that the requirements are being executed into the system. And *requirements validation* detects and fixes errors if found (Ville T, Casper L, Daniela D and Maria P, 2015). *Requirements management* produces bassline requirements and monitoring them. This process involves participation from the stakeholders (Bahiya M and Abdelhamid M, 2015). Requirements development process are summarized as:

- *Requirements Elicitation* occurs in the initial stage. This stage consists gaining knowledge of what is needed and identifying the requirements. And the information gathered needs to be provided to the development team (Westfall L, 2005) (Sean I, 2001).
- *Requirements Analysis* happens directly after the previous stage. This stage involves "cleaning up" the gathered requirements. This is to ensure that they are consistent, complete and viable (Karl W, 2003).
- *Requirements Specification* is the third stage in the requirements development process. This is the process where all the refined requirements are formally documented (Westfall L, 2005).
- *Requirements Validation* usually is performed at the end of the requirements development process. It is done to confirm the requirements are consistent, adequate, adjustable, unambiguous, concise, quantifiable, testable and traceable (Westfall L, 2005).

- *Requirements Management* is the stage consists of activities involving conducting analysis on the change requested to assess their impact. Based on the completed analysis, the implementation of the changes will be decided, either to approve or disapprove (Westfall L, 2005).

However, in ASD, RE process can be challenging because performing RE takes time which could be spent writing codes (Kassab M, 2014). Authors such as (Glinz, 2005) and (Mabrok et al, 2015) categorize them into:

1. **Functional requirements (FRs)** whereby the purpose of the system is defined (Faisandier, 2012). Functional testing's checks whether the software's required behavior matches the specifications set.
2. **NFR (NFRs)** where the performance characteristics of the system is judged. This is important because it determines how the system performs and describes the aspects that affects how well can the system function (Mabrok et al., 2015).

Table 2.3 below lists down the difference between the two types of testing.

Table 2. 4 Difference between Functional Testing and Non-Functional Testing

	Functional Testing	Non-Functional Testing
Execution Period	Functional testing is executed before non-functional testing	Non-functional testing is executed after functional testing
Focus Area	Based on customer's requirements	Focuses on customer's expectation
Requirements	Easy to define requirements. Carried out using the functional specifications	Difficult to define requirements. Carried out using the performance specifications
Goal	To validate the system behaviour against the specification	To validate the performance and technical aspect of the system
Functionality	Describes what the system does	Describes how the system works
Test Example	Check the login functionality	The login page should load in 2 seconds
Testing Types	• Unit testing • User Acceptance • Integration Testing • Regression testing • White Box Testing • Black Box Testing • Interoperability	• Performance Testing • Security Testing • Scalability Testing • Usability Testing • Portability Testing • Reliability Testing • Compliance Testing

Defined by IEEE, NFR is "a software requirement that describes not what the software will do, but how the software will do it". Stakeholders actually care about NFRs, because it decides the quality characteristics of the system (Kiran K and Arvind K, 2013). These authors emphasize that NFR plays a big part in RE. It may be the deciding factor of a systems success or disaster. There's a number of definitions of NFR provided by various researchers, Glinz has provided a survey of all the definition of NFR in the last 20 years. Table 2.4 contains the definitions of NFR from different authors (Glinz M, 2007).

Table 2. 5 Definitions of NFR from Various Authors

Author	Definition
(Anton A, 1997)	Concentrates on the non-behavioural characteristics of the system, gathering the properties and limitations on which a system must function.
(Davis A, 1993)	The overall characteristics of the system containing portability, reliability, efficiency, human engineering, testability, understandability, and modifiability.
(Jacobson I, Booch G and Rumbaugh J, 1999)	Extracts the physical constraints on a functional requirement. Specifies the system attributes such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability.
(Kotonya G and Sommerville I, 1998)	Categorized as the requirements that does not concern the functionality of the system. The external constraints that the product needs to meet that is extracted from restrictions on the system being developed and the development process.
(Mylopoulos J, Chung L and Nixon B, 1992)	Focus on how the system should do by gathering requirements such as performance, reliability, maintainability, portability, robustness. There is no complete list of non-functional requirements.
(Ncube C, 2000)	The behavioral attributes that the functions must follow such as performance, usability, etc.
(Robertson J and Robertson S, 1999)	A quality that a product must possess such as speed, appearance or accuracy.
(Wiegers K, 2003)	Attributes that a system has to represent or a rule it has to obey, not necessarily an observable behaviour.

Proper NFRs testing can produce a quality product. Vikas and Ravi lists the common NFRs that developers come across when handling a software system:

- *Performance* is where the processing time of the system is judged. For example, the time needed for a system or page to load, refresh, etc. This is the most common NFR for software developers as the system performance is vital.

- *Reliability* is the system's ability to recover when a failure occurs. For example, what can be the maximum downtime?
- *Availability* is the systems operational time. For example, specific times of the day when the system is available.
- *Compatibility* covers the ability of the system to operate seamlessly on different platforms, either hardware and software or even both. For example, can the system operate on shared applications? Can the system operate on 3rd party applications?
- *Usability* is the measures how hard is it to use the system. For example, the ease of use, how many languages it supports, are the colors confusing, etc.
- *Maintainability* shows how easy can the system be modified or undergo some changes. This involves adding or removing functionalities or even bug fixing.
- *Interoperability* is the ability to work with other systems with no limitations and difficulty in the access aspect.
- *Recovery* shows how the system gets back up and function as per normal when there's an issue or when it encounters damage. For example, how long does it take for the system to get up and functioning at its original state.
- *Robustness* shows the toughness and how the system handles issues that come up during execution. For instance, can the system still operate if wrong inputs were inserted?
- *Resilience* is the ability of the system to maintain its standard to function normally even if errors are encountered.

Author Mairiza conducted an analysis on numerous types of systems and applications based on three scopes of NFRs. The first scope is the *definition and terminology*. The second scope is followed by the *types*. Finally, the third scope covers the *relevant NFRs.* Table 2.5 provides the findings from the analysis (Mairiza D, 2010).

Table 2. 6 Relevant NFRs on Different Application Domains

Application Domain	Relevant NFRs
Banking and Finance	Accuracy; Confidentiality; Performance, Security; Usability
Education	Interoperability; Performance; Reliability; Scalability; Security; Usability
Energy Resources	Availability; Performance; Reliability; Safety; Usability
Government and Military	Accuracy; Confidentiality; Performance; Privacy; Provability; Reusability; Security; Standardizability; Usability; Verifiability; Viability
Insurance	Accuracy; Confidentiality; Integrity; Interoperability; Security; Usability
Medical/ Health Care	Communicativeness; Confidentiality; Integrity; Performance; Privacy; Reliability; Safety; Security; Traceability; Usability
Telecommunication Services	Compatibility; Conformance; Dependability; Installability; Maintainability; Performance; Portability; Reliability; Usability
Transportation	Accuracy; Availability; Compatibility; Completeness; Confidentiality; Dependability; Integrity, Performance; Safety; Security; Verifiability

Although there are many types of NFR, this paper will only focus specifically on the performance and security aspect in agile development.

Performance testing is performed to validate a system's technical attributes. These attributes consist of the system's speed, stability or scalability (Meier J, Farre C, Bansode P, Barber S and Rea D, 2007). The system is tested using parameters such as access time, execution time, load time, run time, the degree of success, the degree of failure, downtimes, the system's reliability as a whole, etc. (Itti H and Rajender S, 2015). Mainly, performance tests are conducted to ensure the system adheres to the performance objectives set. Cristina Sabrina and Daniela lists the three types of testing approaches in order to achieve a set of goals (R. Cristina, M. Sabrina and S. Daniela, 2016):

- *Performance Test* is performed to examine the system's responsiveness, stability, scalability, reliability or speed. Prior to the testing, the system should have business goals set to check if the set goals are achieved.
- *Load Test* checks how the system functions when exposed to different load conditions. Analyzes the system's ability to handle heavy load volumes.

- *Stress Test* pushes the limit of the system by placing it under huge load conditions to test the endurance. It also checks how will the system recover from an event of a failure.

Security testing is considered to be a critical to a system as it checks how secure is the system and reduces vulnerabilities of being attacked (Warren C, 2011). The security vulnerabilities include data protection, integrity, authorization, confidentiality, availability and authentication issues (Crispin L and Gregory J, 2014). The priority for security testing may vary across different domains. For example, it will be key for companies that are in the finance domain such as banks etc. Attackers are ever ready to pounce on a vulnerability; vulnerability is considered as an error in the system (Verndon D and McGraw G, 2004). Depending on system, there are several types of security testing that can be used such as: Vulnerability Assessment and Penetration Test. Vulnerability Assessment is conducted to identify security weaknesses or flaws of the system that will be able to provide access to attackers. This method is list-orientated meaning the testing is done in categories based on their importance. On the other hand, penetration testing is done after vulnerability assessment. This activity is conducted to simulate a real attack scenario to test the robustness of the system. By doing so, security loopholes can be determined without inflicting any damage to the system. Testers are provided the authority to attempt to defeat the system and find faults, this is all done in with authorization (Jai N and Mehtre B, 2015).

2.5 Factors influencing the testing of NFR

In this section, we review the factors influencing the testing of NFR from findings of different authors.

In the year 2012, authors Vikas Bajpai and Ravi Prakash Gorthi mentioned that there has been a hike in research made on the field of NFR, however there are still issues when it comes to measuring or specifying these requirements. During the software development process, NFR are rarely considered to begin with. The factors for neglecting NFR are:

- *Priority*. Functional requirements are considered sufficient while NFR are ignored.
- *Lack of tools*. Limited techniques and tools available to help determining the requirements.
- *Awareness.* The lack of understanding and knowledge on how non-functional testing can make a significant impact or improvement on a system.
- *Culture.* Ideology of considering non-functional testing as a crucial aspect of a software development process.

Standish Group (2014) published a study focusing on factors causing software project failures. A total of 8 factors were identified and 5 were related to RE. The 5 factors are lack of participation from customers, incomplete requirements, constant amendments to the requirements, unrealistic expectations and also numerous not needed requirements identified.

- *Low customer involvement*. Developers mindset may differ to customer mindset which can cause incorrect prioritization of requirements.
- *Incomplete requirements*. The process of requirement analysis not conducted in detail.
- *Changes in requirements*. As the project develops, new requirements can emerge causing unexpected changes in the whole project.
- *Unrealistic expectations.* Lack of communication may lead to one end being dissatisfied with the system as requirements may differ.
- *Unnecessary requirements identified*. Some requirements are more critical compared to another which is why prioritization of requirement is key.

Authors Bahiya M. Aljallabi and Abdelhamid Mansours (2015) state that although agile methodology plays an important part in improving the software development process, there's numerous limitations as soon as it boils down to requirement analysis. NFR is often neglected during the requirement analysis process. The factors for neglecting NFR

are minimal documentation, lack of communication with customer, project budget and limited time.

- *Minimal documentation*. There are no templates or past documentation to assist in new projects.
- *Communication with customer*. Incorrect prioritization of requirements can lead to a lot of rework if communication with customers is unclear.
- *Budget/Cost*. Budget estimation is never fixed as requirements change.
- *Time*. Lack of time is usually an issue in an agile development process. No proper time is allocation for non-functional testing.

In the year 2015, Ville T. Heikkil¨a, Casper Lassenius, Daniela Damian and Maria Paasivaara collected information from articles reporting problems related to agile RE approach. The issues were organized into 6 themes: no proper communication between clients or customers, the uncertainty in user story format, requirements prioritization not figured out thoroughly, an increase in technical cost, high knowledge dependence on implicit requirements and inaccurate evaluation on energy aspect.

- *Communication issues with clients.* Lack of communication with customer or customer representative leads to incorrect prioritization of requirements. This is because developers might not understand how the business aspect works and make wrong decisions.
- *User story format unclear.* User stories often lack clear explanation on the design process of the software and requirements may need to be divided accordingly. NFRs are often ignored.
- *Issues in prioritizing requirements.* There are several reason why prioritizing requirements is not an easy task in agile development.
    - Agile development method does not have a phase to analyse requirements causes lack of proper requirements
    - Requirement prioritization is usually related to business value which will hide the system improvement related requirements
    - Conflicting requirement needs between customers

    - Unrealistic customer expectations
    - Clients unable to decide on requirement prioritization
- *Increasing technical debt.* The system architecture might be compromised due to the short period of time provided. A complete re-write of the system architecture might be required.
- *Reliance on tacit requirements knowledge.* Determining requirements requires a set of skill, experience and understanding on the importance of requirements analysis.
- *Cost.* Due to the fact that requirements were not properly listed, the project may encounter various unexpected issues with the system. This will add to the project's expenses.
- *Time.* Due to the fact that requirements were not properly listed, the project may encounter various unexpected issues with the system. This will lengthen the project's completion date.

The study also includes findings of solutions to several issues above. Table 2.6 lists the solutions.

Table 2. 7 Solutions established to handle the issues

Issues	Solutions
• Communication	- Traditional requirements engineering role - Clear requirements elicitation - Additional requirements documentation - Extending automated testing ideas to requirements validation
• User story format unclear	- More details to be included in user story format - Limiting the use of user story format for requirements elicitation
• Reliance on tacit requirements knowledge	- Additional requirements documentation to help the process and other members

In recent times, Cristina, Sabrina and Daniela (2016) conducted interviews on agile team members to determine the influencing factors of conducting testing of NFR. Based on the results obtained, they were able to organize the data into the following factors:

- *Priority.* To find out how will the system benefit from performing non-functional testing. It was identified as the main factor. However, the priority depends on different aspects such as:
    - *System Characteristic.* This includes 1) the type of system; 2) user experience on the system; and 3) trend analysis of the system.
    - *Project Type.* This means whether it a new development of a system or changing/fixing an existing system.
    - *Criticality to Business.* This is based on client or the business expectations and the impact to the system if non-functional testing is conducted
- *Time Pressure.* Time is always a factor in an agile environment. Functional testing always come first compared to performance testing due to time constraints. However, it was found that time is always provided for security testing due to its criticality to a system.
- *Cost.* It is discovered that the cost factor has two different views. 1) The cost of failure; this is the additional cost if the system fails the non-functional testing. 2) Budget; where the clients decide not to allocate a budget for the testing because they lack the knowledge in knowing how can the testing benefit the system.
- *Technical Issues.* This is when the issue lies on the system codes. In this case, there are 3 categories.
    - *Production Incidents.* If there's a real issue during the production, only then non-functional testing will be considered.
    - *Resource Utilization.* Analyzing and conducting an assessment on the performance of the system, then decide if the testing for NFRs is needed.
    - *Environment.* Performing tests in an inappropriate environment size that does not provide accurate results
- *Awareness.* The lack of knowledge on how non-functional testing and benefit a system significantly. They believe that if the system does what it is supposed to do, then the system is fine.

- *Culture.* Businesses and developers need to create the habit of considering non-functional testing for a system. Agile developers specifically should always take both FR and NFR testing into consideration in the development stage.
- *Experience.* Due to bad past experiences, the senior members in the team are usually conscious in need of non-functional testing. However, the younger team members usually tend to tie down their focus on the system's functional aspects. Other than that, the testing of NFR requires a set of skill or expertise to ensure proper testing is conducted.

Table 2.7 shows the four practices created for the team members to tackle the above factors influencing non-functional testing.

Table 2. 8 Practices established to handle the factors

Factor	Practices
• Priority	- Discuss in detail on non-functional aspects - User story to be developed - Senior team members to emphasize the importance of non-functional testing
• Awareness • Culture	- Non-functional requirements and testing need to be reviewed by at least one member from different roles (developers, testers, software architects and product owners)
• Priority • Culture • Cost • Technical Issues • Awareness • Time Pressure	- Communication between developers and testers are to be done clearly - Communication with other teams and real users should be practiced as well to have different views
• Priority • Awareness • Culture	- Quality mindset is required throughout the whole process - Understanding of the importance of non-functional testing is required

2.6 Summary of factors influencing the testing of NFR

After reviewing the factors influencing the testing of NFR by various authors, the summary of the findings is tabulated in Table 2.8. Paper 1 & 2 does not limit to only ASD. Paper 2 & 3 consist of share factors that affect the RE process in general. As the table shows, there are a number of repeating factors among different studies.

Table 2. 9 Summary of the factors influencing NFR testing

Paper	1	2	3	4	5
Author	Vikas Bajpai and Ravi Prakash Gorthi (2012)	Standish Group (2014)	Bahiya M. Aljallabi and Abdelhamid Mansours (2015)	Ville T. Heikkil¨a, Casper Lassenius, Daniela Damian and Maria Paasivaara (2015)	Cristina, Sabrina and Daniela (2016)
Methodology	General	General	Agile	Agile	Agile
Non-functional requirement (NFR) / Functional requirement (FR)	NFR	NFR , FR	NFR	NFR , FR	NFR
Factors	• Prioritization of requirements • Lack of tools to define requirements • Awareness • Culture	• Lack of communication with customer • Prioritization of requirements • Cost/Budget	• Lack of communication with customer • Minimal documentation • Time pressure • Cost/Budget	• Lack of communication with customer • User story format unclear • Prioritization of requirements • Reliance on tacit requirements knowledge • Time pressure • Cost/Budget	• Prioritization of requirements • Time pressure • Cost/Budget • Technical issues • Awareness • Culture • Experience

2.7 Summary

This chapter covered the previous studies on the definitions, ASD process, a comparison summary of the traditional methodology against agile methodology, the agile software testing process, the difference between the two, a detailed review on NFR and the existing studies on the factors influencing NFR testing. Furthermore, we have included a summarize table for the factors influencing NFR testing identified by previous studies.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter describes the methodology implemented. The purpose of this study is to determine the influencing factors of conducting non-functional testing in ASD. We highlight the design of the research steps utilized to achieve our objective.

3.2 Research Design

This design of this research is constructed based on two context which are analytical review and quantitative survey. The first approach is constructed based on the three goals identified for this research, which are: (1) To identify the influencing factors of testing of NFR in ASD, (2) To discover the challenges faced to conduct the testing of NFR in ASD and (3) To determine the practices that can be adopted by agile team member when conducting non-functional testing. We reviewed the literature focusing on NFR testing and extracted the relevant influencing factors of conducting the testing of NFR in agile environment and a few non-agile environment software development processes mentioned in the analyzed papers in Chapter 2.

In addition to that, a quantitative survey was deployed as the second approach to empirically evaluate to identify the influencing factors of conducting NFR testing in ASD environment, to discover the challenges that the agile team members run into when conducting the testing and to determine the guidelines adopted by agile team members to get passed the obstacles when conducting non-functional testing. By using Cristina,

Sabrina and Daniela's (2016) interview guide as a bassline, a set of questions was built to collect the relevant information to fulfill the research's objectives. Other than that, to validate the survey and ensure the collected data is what we require to achieve the objectives, an expert review was performed.

The Figure below shows the planned research approach.

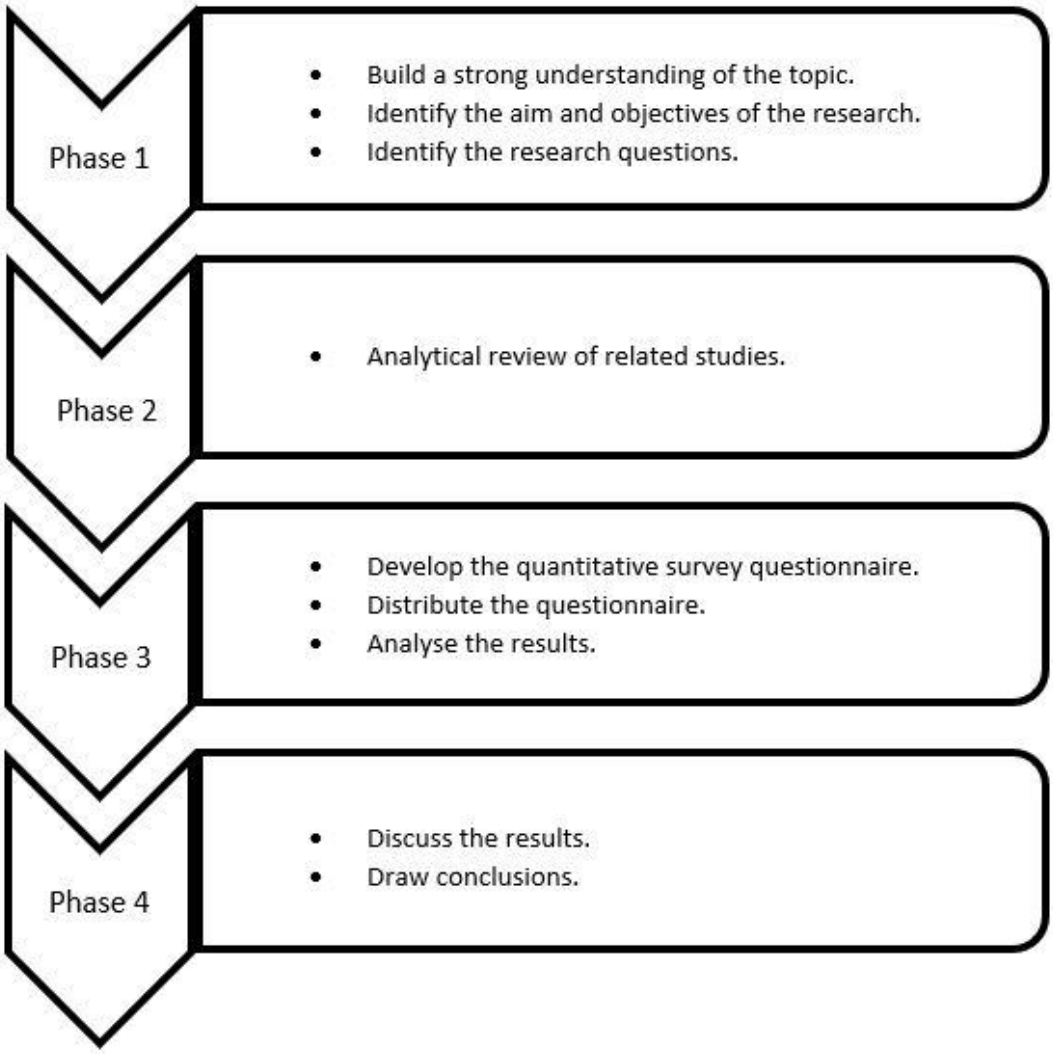

Figure 3. 1 Research Approach

3.3 Expert Review

Reliability of a study is important as well as the validity but those will only come when information is accurate, trustworthy and credible (Davies & Dood, 2002). An expert review was conducted because we strive to have clear statements in the set of questions and most importantly the participants understands the questions how we want them to. A total of four experts were brought in to perform the review and the data collected from those four experts was left out and not taken into the total count of the sample of the study. Two of them were from educational background whereas the other two were from private sectors. The two experts from the educational background consisted of one male and one female, both associate professors in their late-30's. The remaining two experts from the private sector were both male; software testers in their mid-30's. We looked into their feedback on the statements prepared. Suggestions were taken and used to better the questionnaire but no major changes were required.

3.4 Data Collection

We were collecting the data between the month of July and August 2019. Google Forms is where the survey was created and the data were primarily gathered from the online source. Printed versions of the survey were distributed as well. The survey incorporated a small write up about non-functional testing in ASD.

For this study, we chose questionnaire as the data collection method. It's a set of questions or statements prepared to get hold of information or data from participants; whereby questions are usually close-ended (Kothari, 2004). Questionnaires don't cost much for the ability to gather large amount of sample in a desired group of people (Akbayrak, 2000). Instead of interviews, we chose to do a set of statements built by ourselves to ensure we

get the data required whereas interview can go off course and lead to collection of unnecessary or excessive data due to interviewer's skills (Phellas, Bloch & Seale, 2011).

Overall, we had 139 participants from diverse backgrounds. However, only 128 responses were extracted and found useful. The Google Form survey link that was used to collect the responses is stated below:

http://bit.ly/ashwinesh

3.4.1 Survey Sample

The size of the sample used was determined based on the need to gather sufficient statistical power. We had a timeframe set for data collection and all the valid responses received were used as results. A combined total of 139 responses for the questionnaires was collected from online and physical surveys but only 128 responses were extracted as useable results whereas the remaining responses were discarded due to invalid responses.

3.4.2 Research Instrument

This research had two ways of completing the survey which is an online copy or a physical copy of it (found in Appendix A) which aided as an instrument to gather information from the respective participants. The structure of the survey is as follows:

a) Demographic Information

This is the first section of the survey which consists of eight questions which is regarding the respondent's background.

b) Perception on Non-Functional Testing in Agile Software Development

31 statements that are relevant to NFR testing in ASD, the factors influencing non-functional testing, what are the challenges faced when conducting non-functional testing

and the practices used to overcome obstacles when conducting non-functional testing that were identified from the existing guidelines are found in this section.

3.5 Quantitative Approach

Quantitative research approach involving calculation or measurement to be able to use the information statistically (Leedy, 1993). The main goal of this method is to gather measurable data from respondent's perspective on understanding of subject, issues and views of the topic. A quantitative researcher will always gather and validate the data obtained very carefully; commonly in a form or integers to be able to translate the to be able to view it statistically in a computer (Neuman, 2006). Quantitative research allowed us to quantify the gathered information by generating numerical data that transformed into useable statistics for our study.

3.5.1 Questionnaire Development

Part (1) Demographic Information consists of eight questions to get an idea of the participants background. Part (2) consists of 31 statements that are related to NFR testing in ASD environment, the factors influencing non-functional testing, what are the challenges faced when conducting non-functional testing and the practices used to overcome obstacles when conducting non-functional testing which were identified from the existing guidelines. Expert review was conducted to validate the questions and statements.

3.6 Data Analysis

We opted to use SPSS version 26 as our data analysis tool for this research. Descriptive analysis was conducted to identify the demographic information of the respondents. Cronbach's Alpha reliability test was also conducted to measure the reliability of the statements presented in the questionnaire. Furthermore, Kaiser-Meyer-Olkin (KMO) test was used to measure if the data is suitable for Factor Analysis. Bartlett's Test was performed to compare the correlation matrix to the identity matrix; this checks for redundancy between variable that can be included together with certain factors. Last but not least, Exploratory Factor Analysis (EFA) was performed to find how the variable's measured relate to one another.

3.7 Summary

We illustrated the methodology implemented and explained why the method was chosen to be used in this research. We also explained the design structure of the research and the steps taken to conduct it. Two prominent approaches were used in this study; the first is analytical review of the existing factors and secondly, quantitative survey. The data collection method, research instrument and details of the data analysis were also highlighted in this chapter.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Introduction

This chapter highlights the results of a quantitative survey that was conducted to identify the factors affecting the non-functional testing in agile software environment. The data obtained from the 128 participants were analyzed using various statistical tests. The detailed report of the results is elaborated.

4.2 Descriptive Information of Respondents

The sample consisted of 128 participants from various backgrounds living in Malaysia (Mean Age= 32, Standard Deviation = 6.71). The following section presents the demographic information obtained from the survey.

4.2.1 Gender

Figure 4.1 shows the graphical representation of the respondents' gender. Based on the data, we obtained a good balance of the distribution between male and female respondents in this study, even though there were slightly more males (55.47%) compared to females (44.53%) in the sample.

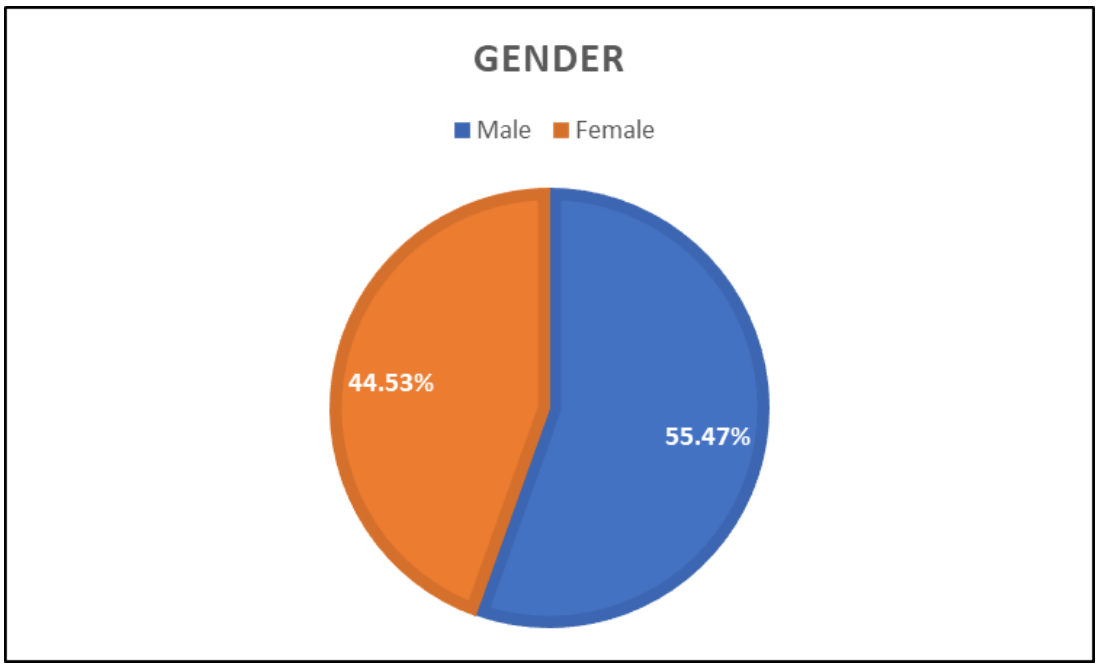

Figure 4. 1 Gender

4.2.2 Education Background

Figure 4.2 presents the graphical representation of the respondents' educational background. The majority of the respondents (96.88%) were clearly from the tertiary level completion category, which is 124 of them from the total respondents. The remaining were divided between secondary and primary level with 2.34% and 0.78% respectively.

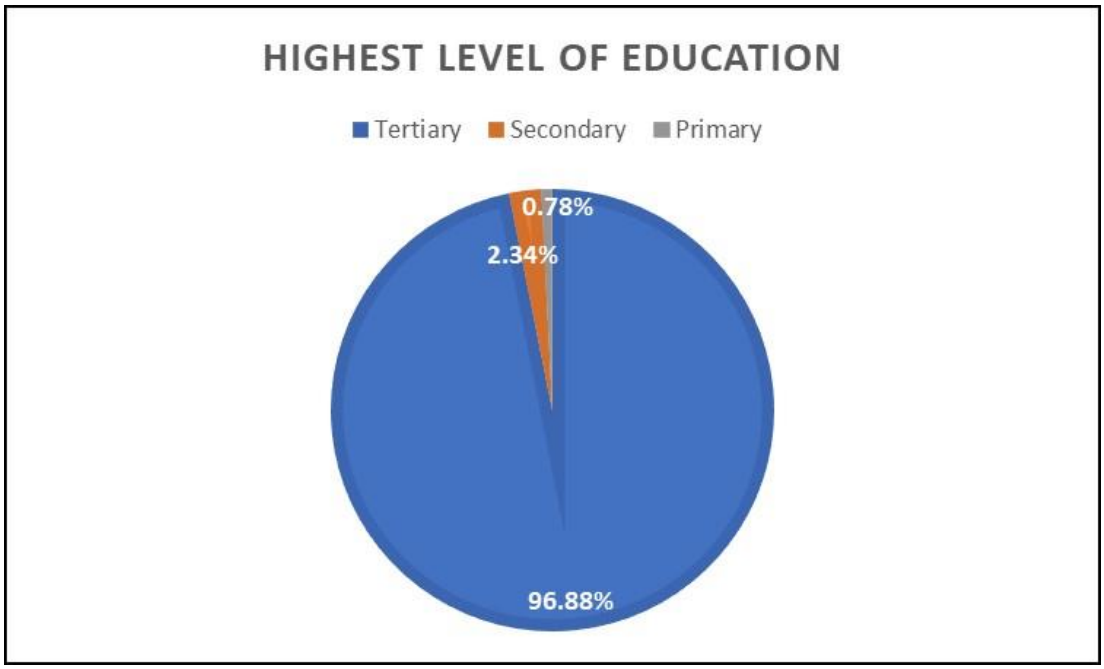

Figure 4. 2 Education Background

4.2.3 Knowledge on Agile Software Development

The respondent's knowledge on ASD is presented in Figure 4.3. presents the graphical representation of the respondents' educational background. The chart displays a wide number of the respondents (76.56%) had prior knowledge on ASD. It was gathered that 23.44% of the respondents had no knowledge on ASD.

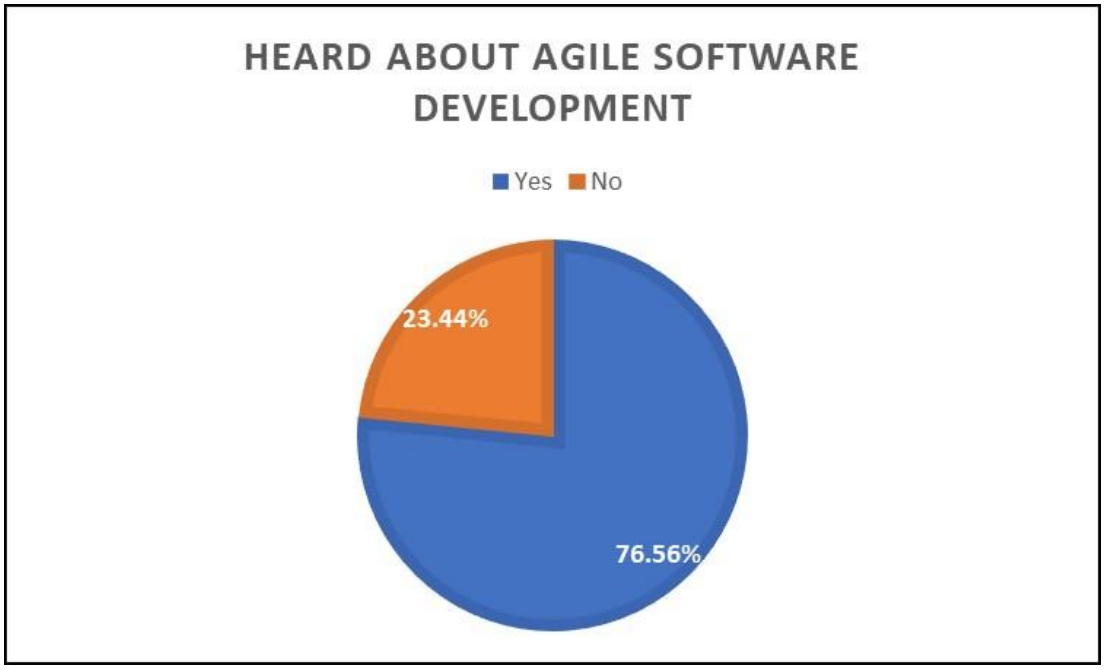

Figure 4. 3 Knowledge on ASD

4.2.4 Involvement in Software Development Process

In Figure 4.4 we are able to see that a total of 78 respondents out of 128 (60.94%) have an exposure in the field of software development process whereby the remaining percentage of 39.06 have no particular involvement in the process.

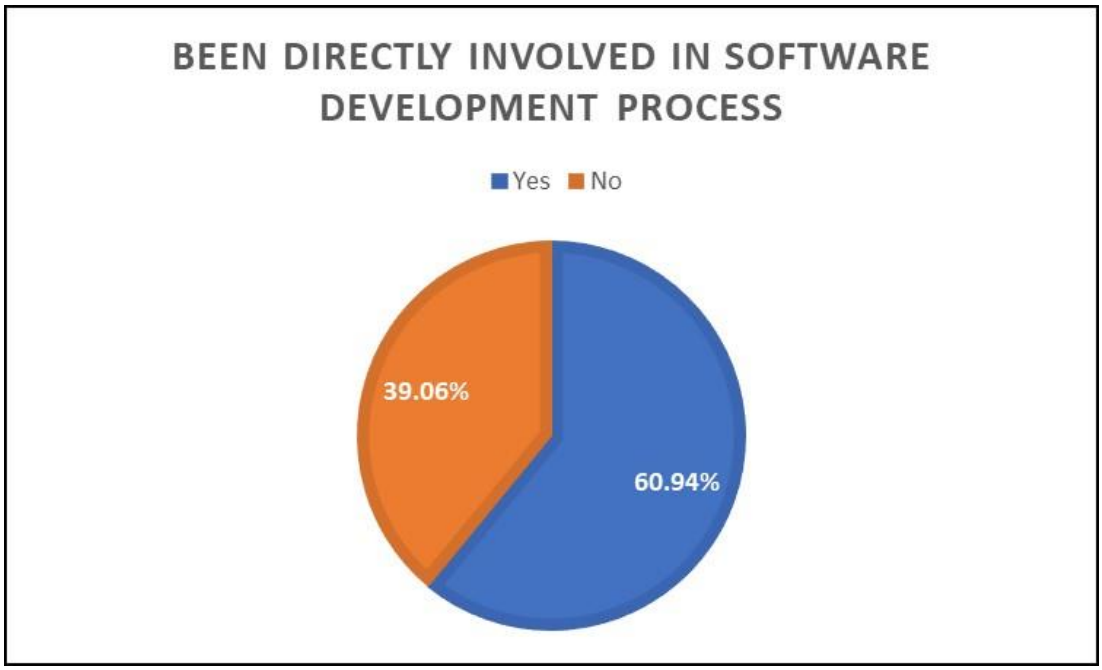

Figure 4. 4 Involvement in Software Development Process

4.2.5 Knowledge on Software Testing

In Figure 4.5, we present the graphical representation of the participants' knowledge on software testing. Most of the participants (78.91%) have heard about software testing process. Only a small portion (21.09%) have no knowledge on software testing.

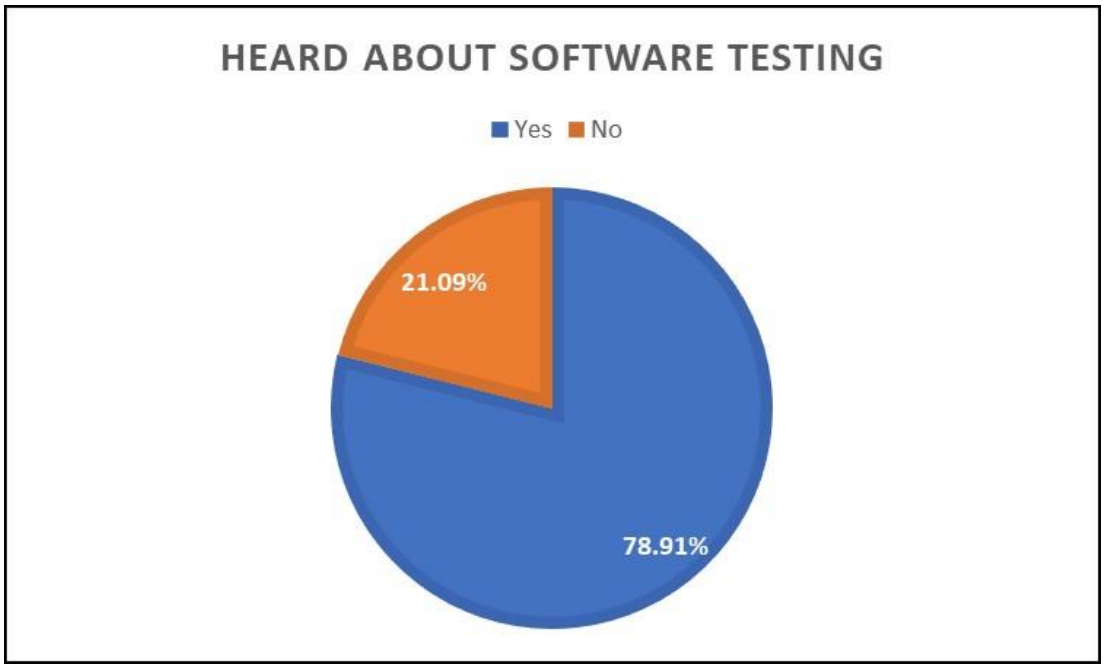

Figure 4. 5 Knowledge on Software Testing

4.2.6 Involvement in Testing of Non-Functional Requirements

In Figure 4.6, the graphical representation of the respondents' involvement in testing NFR is shown. The chart shows an almost even split of respondents. A percentage of 53.91 respondents have been involved in NFR testing. However, the remaining respondents (46.09%) have not been involved in the process of testing NFR.

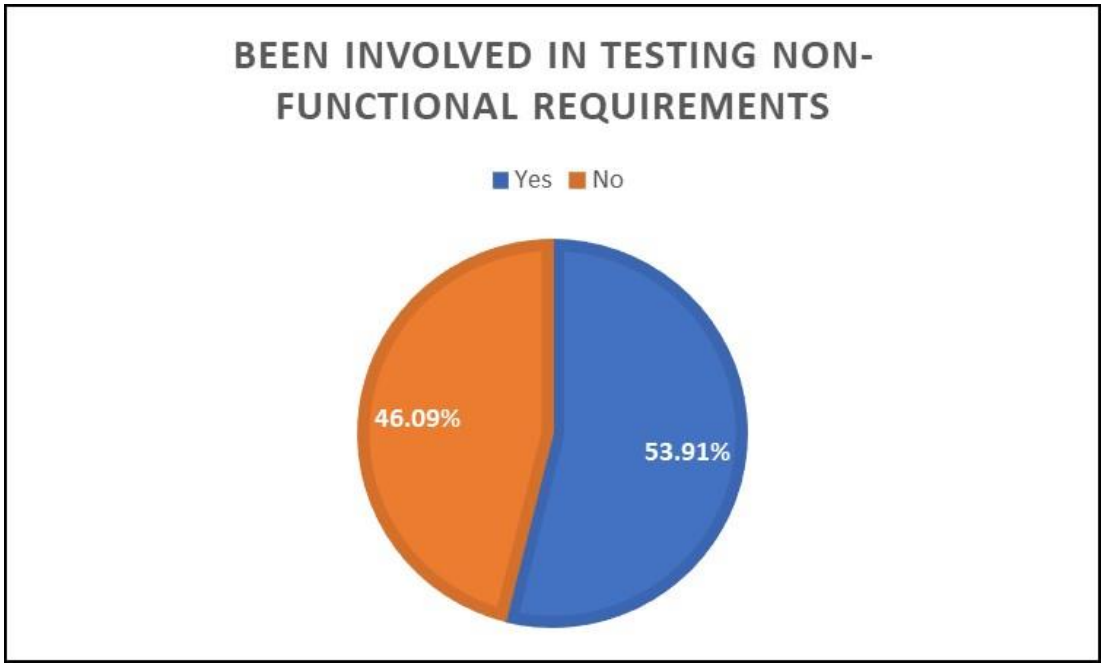

Figure 4. 6 Involvement in Testing NFR

4.2.7 Work Experience

Figure 4.7 provides an idea of the amount of work experience a participant have in a software development environment. It is found that a chunk of the respondents (39.06%) belong to the category without any work experience in the software development area. The rest of the respondents have acquired at least a minimum amount of work experience in software development field. Majority of them (28.90%) were from the category of less than 5 years of experience. Next were the category between 5 to 10 years with 15.62% of the overall. The remaining were split between the categories 11 to 15 years and more than 15 years with 12.50% and 3.91% respectively. Only a handful of respondents had the experience of more than 15 years in this field.

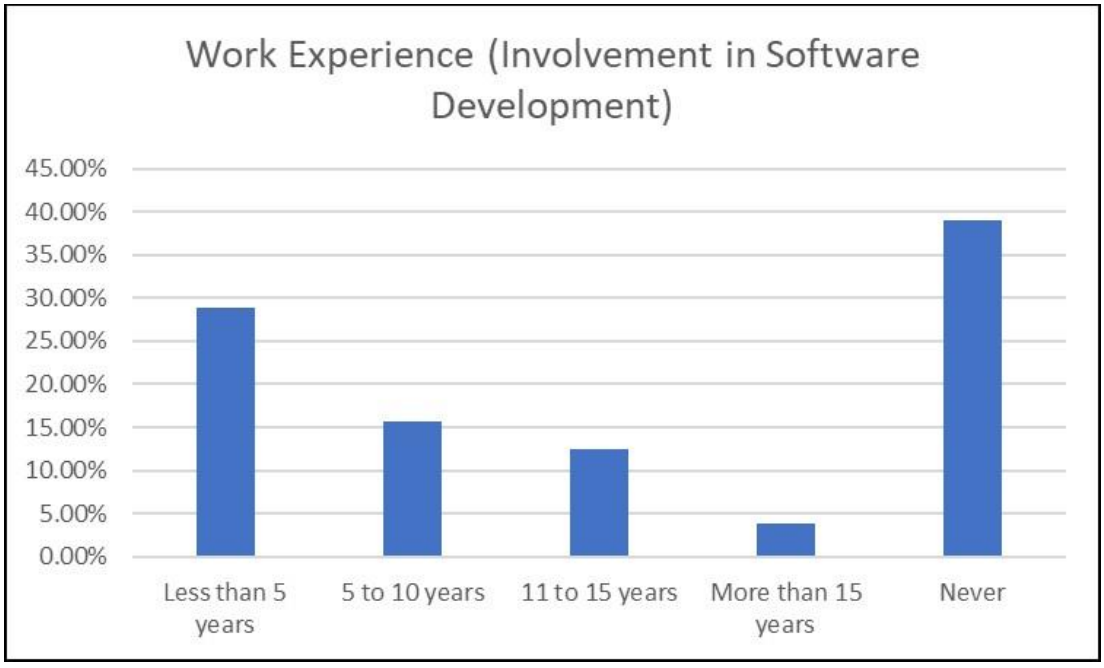

Figure 4. 7 Work Experience

4.2.8 Summary of the Respondents' Demographic Information

Table 4.1 provides a summary of the participants' demographic information presented above.

Table 4. 1 Descriptive Information of Respondents'

Characteristics	n	%
Gender		
Male	71	55.47
Female	57	44.53
Highest Level of Education		
Tertiary	124	96.88
Secondary	3	2.34
Primary	1	0.78
Heard about agile software development		
Yes	98	76.56
No	30	23.44
Been directly involved in software development process		
Yes	78	60.94
No	50	39.06
Heard about software testing		
Yes	101	78.91
No	27	21.09
Been involved in testing non-functional requirements		
Yes	69	53.91
No	59	46.09
Work Experience (Involvement in Software Development)		
Less than 5 years	37	28.90
5 to 10 years	20	15.62
11 to 15 years	16	12.50
More than 15 years	5	3.91
None	50	39.06

4.3 Descriptive Statistics Representation

We present the individual variable's descriptive statistics in detail in this section. The mean score statistic interpretation was conducted based on Table 4.2 which was adopted from Siron, Tasripan & Majid (2013).

Table 4. 2 Mean Score interpretation

Mean Score	Interpretation
1.00 – 1.80	Strongly Disagree (SD)
1.81 – 2.60	Disagree (D)
2.61 – 3.40	Moderate Agree (MA)
3.41 – 4.20	Agree (A)
4.21 – 5.00	Strongly Agree (SA)

Table 4.3 illustrates the mean score value of *ASD Methodology*. As shown in the table, the participants strongly agreed on one item whereas the other two items scored agree. The overall mean score value of *ASD Methodology* is 4.26 (Agree).

Table 4. 3 Mean Score for ASD Methodology

Items	Mean	Std. Deviation	Status
I understand there are various types of software development methodologies available.	4.22	0.639	SA
In software development process, agile methodology is widely used in projects.	4.20	0.732	A
I prefer agile methodology compared to the traditional methodology.	4.15	0.620	A
Overall mean score value of Agile Software Development Methodology			4.19
Overall mean status of Agile Software Development Methodology			A

Table 4.4 illustrates the mean score value of *Software Testing (Non-Functional Testing)*. The participants strongly agreed on four out of the five items whereby the remaining one item scored agree. The overall mean score is 4.25 (Strongly Agree).

Table 4. 4 Mean Score for Software Testing (Non-Functional Testing)

Items	Mean	Std. Deviation	Status
It is important to conduct software testing on every project.	4.31	0.599	SA
Non-functional testing is more important than Functional testing.	4.18	0.539	A
Non-functional testing is not taken seriously by testers.	4.26	0.578	SA
Non-functional testing is required in all projects.	4.21	0.556	SA
If provided a choice, I would choose to conduct non-functional testing in all projects.	4.27	0.582	SA
Overall mean score value of Software Testing (Non-Functional Testing)			4.25
Overall mean status of Software Testing (Non-Functional Testing)			SA

Table 4.5 illustrates the mean score value of *Non-Functional Testing Process.* The respondents strongly agreed on three out of the four items whereby the remaining one item scored agree. The overall mean score value is 4.23 (Strongly Agree).

Table 4. 5 Mean Score for Non-Functional Testing Process

Items	Mean	Std. Deviation	Status
Non-functional testing process is started when the project begins.	4.20	0.562	A
Non-functional testing process should start as soon as a project begins.	4.24	0.572	SA
Non-functional testing is done by anyone in the team.	4.22	0.560	SA
Non-functional testing should only be done by a specialist.	4.27	0.585	SA
Overall mean score value of Non-Functional Testing Process			4.23
Overall mean status of Non-Functional Testing Process			SA

Table 4.6 illustrates the mean score value of *Factors influencing Non-Functional Testing*. The respondents strongly agreed on ten out of the thirteen items presented whereby the remaining three items scored agree. The overall mean score value is 4.24 (Strongly Agree).

Table 4. 6 Mean Score for Factors influencing Non-Functional Testing

Items	Mean	Std. Deviation	Status
Time constraint.	4.28	0.574	SA
Budget constraint.	4.27	0.585	SA
Failing to prioritize in the initial stage.	4.21	0.527	SA
Technical issues.	4.20	0.548	A
Awareness of the importance.	4.24	0.529	SA
Culture of the company.	4.23	0.550	SA
Experience of the members.	4.24	0.572	SA
Lack of communication with customer.	4.19	0.572	A
Minimal documentation of the process.	4.23	0.564	SA
Incorrect individual performing the tests.	4.35	0.875	SA
Over-reliance on manual testing.	4.15	0.875	A
Lack of training invested for team members.	4.25	0.851	SA
Lack of team effort.	4.25	0.910	SA
Overall mean score value of Factors influencing Non-Functional Testing			4.24
Overall mean status of Factors influencing Non-Functional Testing			SA

Table 4.7 illustrates the mean score value of *Challenges Faced When Conducting Non-Functional Testing*. The participants strongly agreed on five out of the total nine items where the remaining four items scored agree. The overall mean score value is 4.21 (Strongly Agree).

Table 4. 7 Mean Score for Challenges Faced When Conducting Non-Functional Testing

Items	Mean	Std. Deviation	Status
I face various challenges in conducting non-functional testing.	4.23	0.564	SA
I am able to overcome the challenges faced.	4.18	0.633	A
Challenges faced in every project are similar.	4.27	0.582	SA
Programmers Writes Tests.	4.19	0.572	A
Lack of documentation throughout the process.	4.25	0.544	SA
Lack of testing skillset between team members to conduct proper testing.	4.21	0.449	SA
Lack of individuals assigned to conduct the testing.	4.19	0.569	A
Requirements are too subjective.	4.22	0.648	SA
Infrastructure overhead.	4.18	0.659	A
Overall mean score value of Challenges Faced When Conducting Non-Functional Testing			4.21
Overall mean status of Challenges Faced When Conducting Non-Functional Testing			SA

Table 4.8 illustrates the mean score value of *Practices to Adopt for Conducting Better Non-Functional Testing*. The participants strongly agreed on six out of the seven items whereby the remaining final item scored agree. The overall mean score value is 4.23 (Strongly Agree).

Table 4. 8 Mean Score for Practices to Adopt for Conducting Better Non-Functional Testing

Items	Mean	Std. Deviation	Status
Non-functional testing requires clear requirements elicitation.	4.26	0.565	SA
Non-functional testing requires additional requirements documentation to help the process and other members.	4.21	0.570	SA
Non-functional requirements and testing need to be reviewed by at least one member from different roles (developers, testers, software architects and product owners).	4.23	0.568	SA
Understanding of the importance of non-functional testing is required.	4.25	0.517	SA
Test planning should be done in the early stage of development.	4.24	0.625	SA
Understanding the proper usage of testing tools to assist with the testing process.	4.21	0.605	SA
Repeat tests multiple times to ensure consistent results.	4.18	0.590	A
Overall mean score value of Practices to Adopt for Conducting Better Non-Functional Testing			4.23
Overall mean status of Practices to Adopt for Conducting Better Non-Functional Testing			SA

In summary, out of the six variables sectioned, five of them produced an overall mean of at least 4.21 and above which resulted in a Strongly Agree status while the first variable, *ASD Methodology*, obtained an overall mean of 4.19 which resulted in an Agree status.

4.4 Reliability Analysis

To conduct the reliability analysis for this study, Cronbach's Alpha test is conducted using IBM SPSS Statistic tool. Cronbach's Alpha is used measure the reliability or internal consistency of a scale, usually when questionnaires contain Likert scales are present (Tavakol & Dennick, 2011). Therefore, to confirm the validity of the questions presented in the questionnaire, Cronbach's Alpha test is conducted. The Table 4.9 below illustrates the range of reliability of the Cronbach's Alpha.

Table 4. 9 Mean Score for Reliability range of Cronbach's Alpha

Cronbach's Alpha	Internal Consistency
$0.9 \leq \alpha$	Excellent
$0.8 \leq \alpha < 0.9$	Good
$0.7 \leq \alpha < 0.8$	Acceptable
$0.6 \leq \alpha < 0.7$	Questionable
$0.5 \leq \alpha < 0.6$	Poor
$\alpha < 0.5$	Unacceptable

However, a high degree of internal consistency should not be solely decided based on high coefficient alpha. This is due to the fact that the size of the test also affects the result. A shorter test length will reduce the value of alpha (Klein, 2008). Therefore, Table 4.10 shows the degree of alpha value when all the 41 items are combined whereas Table 4.11 displays the degree of alpha when the analysis is conducted on items per variable.

Table 4. 10 Overall Cronbach's Alpha

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	Number of Items
.985	.984	41

Table 4. 11 Independent Cronbach's Alpha

Variable	Number of Items	Cronbach's Alpha	Status
Agile Software Development Methodology	3	.721	Acceptable
Software Testing (Non-Functional Testing)	5	.960	Excellent
Non-Functional Testing Process	4	.965	Excellent
Factors influencing Non-Functional Testing	13	.980	Excellent
Challenges Faced When Conducting Non-Functional Testing	9	.910	Excellent
Practices to Adopt for Conducting Better Non-Functional Testing	7	.952	Excellent

Table 4.10 indicates that Cronbach's Alpha of all the items combined is .985, which is at the 'Excellent' status. Table 4.11 displays the test conducted on independent variable and the status obtain were all above the 'Acceptable' level at least. The results show that the internal consistency and reliability of all the items are more than adequate.

4.5 Kaiser-Meyer-Olkin and Bartlett's Test

Kaiser–Meyer–Olkin (KMO) Test indicates the ratio of the squared correlation between principles to the squared partial correlation between principles. It ranges from the value zero to one where values close to 1 are considered as high values and it indicates that the factor analysis would produce reliable data. When the values are greater than 0.5 are considered to be in the acceptable range (Kaiser, 1974). That being said, the values that fall under 0.5 indicates that the data gathered requires re-work as it will not be useful.

Meanwhile, to test an original correlation matrix to the identity matrix, we conduct the Bartlett's Test. It basically inspects the variable to see if there are any redundancy between

them which can be summarized with certain number of factors. The values beneath 0.05 are highly significant.

Table 4. 12 KMO and Bartlett's Test

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.904
Bartlett's Test of Sphericity	Approx. Chi-Square	8335.023
	Df	325
	Sig.	.000

Table 4.12 indicates that KMO value of all principles are more than acceptable. In fact it falls in the range of superb as the results greater than .9 are taken as superb (Hutcheson & Sofroniou, 1999). The results also indicate that the values of all items are within the acceptable range and indicates that the factor analysis data is useful.

4.6 Exploratory Factor Analysis

Exploratory Factor Analysis (EFA) is a statistical procedure that discovers and explains the correlations between a large set of variables. Researchers use EFA to identify the number of factors influencing the variables in order to analyze which variable 'goes together' (DeCoster, 1998). In this study, EFA was conducted on 41 items related to six variables. Principal component analysis with the use of promax rotation method was used as the extraction method to ensure an understandable output. To simplify the data even more to make it more readable, the items were assigned to shorter codes as shown in Table 4.13 below.

Table 4. 13 Codes and Variables Items

Variables	Statements	Coding
ASD Methodology	I understand there are various types of software development methodologies available.	AS1
	In software development process, agile methodology is widely used in projects.	AS2
	I prefer agile methodology compared to the traditional methodology.	AS3
Software Testing (Non-Functional Testing)	It is important to conduct software testing on every project.	ST1
	Non-functional testing is more important than Functional testing.	ST2
	Non-functional testing is not taken seriously by testers.	ST3
	Non-functional testing is required in all projects.	ST4
	If provided a choice, I would choose to conduct non-functional testing in all projects.	ST5
Non-Functional Testing Process	Non-functional testing process is started when the project begins.	NF1
	Non-functional testing process should start as soon as a project begins.	NF2
	Non-functional testing is done by anyone in the team.	NF3
	Non-functional testing should only be done by a specialist.	NF4
Factors influencing Non-Functional Testing	Time constraint.	FA1
	Budget constraint.	FA2
	Failing to prioritize in the initial stage.	FA3
	Technical issues.	FA4
	Awareness of the importance.	FA5
	Culture of the company.	FA6
	Experience of the members.	FA7
	Lack of communication with customer.	FA8
	Minimal documentation of the process.	FA9
	Incorrect individual performing the tests.	FA10
	Over-reliance on manual testing.	FA11
	Lack of training invested for team members.	FA12
	Lack of team effort.	FA13
Challenges faced when conducting non-functional testing	I face various challenges in conducting non-functional testing.	CF1
	I am able to overcome the challenges faced.	CF2
	Challenges faced in every project are similar.	CF3
	Programmers Writes Tests.	CF4
	Lack of documentation throughout the process.	CF5
	Lack of testing skillset between team members to conduct proper testing.	CF6

Variables	Statements	Coding
Challenges faced when conducting non-functional testing	Lack of individuals assigned to conduct the testing.	CF7
	Requirements are too subjective.	CF8
	Infrastructure overhead.	CF9
	Non-functional testing requires clear requirements elicitation.	PR1
Practices to Adopt for Conducting Better Non-Functional Testing	Non-functional testing requires additional requirements documentation to help the process and other members.	PR2
	NFR and testing should be checked by a person in a different role and should not be done by the same person or a person in the same role.	PR3
	Having the knowledge of the significance of non-functional testing is required.	PR4
	Test planning should be done in the early stage of development.	PR5
	Understanding the proper usage of testing tools to assist with the testing process.	PR6
	Repeat tests multiple times to ensure consistent results.	PR7

Table 4. 14 Pattern Matrix of Exploratory Factor Analysis

Variable / Item	1	2	3	4	5	6
AS1	.851					
AS2	.683					
AS3	.881					
ST1		.843				
ST2		.890				
ST3		.981				
ST4		.951				
ST5		.980				
NF1			.915			
NF2			.975			
NF3			.967			
NF4			.946			
FA1				.838		
FA2				.813		
FA3				.905		
FA4				.967		
FA5				.638		
FA6				.978		
FA7				.939		
FA8				.924		
FA9				.990		
FA10				.735		
FA11				.962		
FA12				.978		
FA13				.966		
CF1					.845	
CF2					.611	
CF3					.951	
CF4					.746	
CF5					.723	
CF6					.744	
CF7					.810	
CF8					.951	
CF9					.917	
PR1						.857
PR2						.909
PR3						.895
PR4						.932
PR5						.874
PR6						.826
PR7						.906

Table 4.15 shows the stats which supports that all the variables remain since they keep most of the original value. No items were removed or merged to cross relation as it is identified they have strong relations among each other. All the loadings values are more than 0.30 which is identified as the minimum desired value. Therefore, the pattern matrix supports all the six variables.

4.7 Summary

Chapter 4 presents the findings of data analysis via three prominent statistical tests. To identify the mean and standard deviation for this study, we conducted a descriptive analysis. Other than that, we conducted a Cronbach's Alpha reliability test to analyzed the relations between a set of items. All the variables achieved Cronbach's Alpha exceeding 0.6, which indicates that the internal consistency is above acceptable. KMO and Bartlett's Test was conducted to identify the convenience in performing factor analysis. The value 0.904 obtained from KMO the test indicates that the items are closely related which means the factor analysis should give reliable variables. As for the Bartlett's Test, the result obtained a value beneath 0.05 which is highly significant. Finally, EFA was carried out to identify the underlying factors where all the six variables resulted in acceptance.

CHAPTER 5

DISCUSSION AND CONCLUSION

5.1 Introduction

In this chapter, we present and discuss the results obtained from the statistical tests conducted to achieve the objectives of this study. Then, we conclude our study by providing suggestions and recommendations for future works.

5.2 Study Overview

The objectives of this study is widen our knowledge on NFR testing on ASD, specifically: (1) to identify the influencing factors and challenges of conducting non-functional testing in ASD, (2) to validate the factors influencing non-functional testing in ASD and (3) to propose the practices that can be adopted by agile team member when conducting non-functional testing.

From previous studies mentioned in Chapter 2, we gathered the identified influencing factors of conducting non-functional testing in ASD, challenges faced by agile team members when conducting ASD and practices adopted by agile team member to conduct proper NFR testing. The items collected were included in the questionnaire developed to validate the items and respondents also included new items to further identify more factors, challenges and practices for this study. The quantitative study involving 128 respondents was carried out to evaluate the items.

5.3 Discussion of Results

The main intend here is to determine the influencing factors of conducting non-functional testing in ASD. Based on the statistical tests done, we have identified all (13) factors were found important. A total of 4 new factors were identified in this study; Incorrect individual performing the tests, Over-reliance on manual testing, Lack of training invested for team members, Lack of team effort. Therefore, all the factors will be accepted and included. The items in the first 3 variables in the results: (1) ASD Methodology, (2) Software Testing (Non-Functional Testing) and (3) Non-Functional Testing process were all found to be important as well as the results supports the significance of NFR testing in ASD, the fact that non-functional testing are not taken seriously by agile team members, non-functional testing process needs to be given focus when at the beginning stage of the project and also the testing should only done by a capable member.

We have discovered 6 challenges that agile team members faced when conducting NFR testing in agile environment. All the challenges were identified from this study. The challenges are; Programmers Writes Tests, Lack of documentation throughout the process, Lack of testing skillset between team members to conduct proper testing, Lack of individuals assigned to conduct the testing, Requirements are too subjective, Infrastructure overhead. The participants agreed to the fact that the do face challenges when conducting the testing but most of the challenges faced are similar from one project to another. The six identified challenges were all confirmed important based on several statistical tests conducted.

We determined a total of 7 practiced that agile team member can adopt to ensure better conducting of NFR testing. A total of 3 were new practices discovered in this study; Test planning should be done in the early stage of development, Understanding the proper usage of testing tools to assist with the testing process, Repeat tests multiple times to ensure consistent results. The seven practices identified were supported by the statistical tests done. The findings obtained will help agile team members to better understand and

conduct non-functional testing in their projects. It can serve as a guide for agile team members to plan ahead and have the knowledge of what to be expected to conduct NFR testing. The practices determined would aid agile team members to build confidence in successfully conducting proper NFR testing in their projects.

5.4 Factors, Challenges & Practices

Table 5.1, 5.2 and 5.3 presents the results to fulfill the objectives of this study. Table 5.1 presents the identified influencing factors of conducting non-functional testing in ASD. A total of 4 new factors were identified in this study; Incorrect individual performing the tests, Over-reliance on manual testing, Lack of training invested for team members, Lack of team effort. Here is the breakdown of all the factors:

1. ***Time Constraint***. Time is always a factor in an agile environment. Functional testing always come first compared to performance testing due to time constraints. However, it was found that time is always provided for security testing due to its criticality to a system.
2. ***Budget Constraint***. The client does allocate the budget for it or developers are told not to conduct the testing due to cost issues.
3. ***Failing to prioritize in the initial stage***. To find out how will non-functional testing add value to the system. It was identified as the main factor. However, the priority depends on different aspects such as system characteristics, project type and criticality to business.
    - *System Characteristic.* This includes 1) the type of system; 2) user experience on the system; and 3) trend analysis of the system.
    - *Project Type.* This means whether it a new development of a system or changing/fixing an existing system.
    - *Criticality to Business.* This is based on client or the business expectations and the impact to the system if non-functional testing is conducted

4. ***Technical issues***. This is when the issue lies on the system codes. In this case, there are 3 categories.
    - *Production Incidents.* If there's a real issue during the production, only then non-functional testing will be considered.
    - *Resource Utilization.* Analyzing and conducting an assessment on the performance of the system, then decide on the need for non-functional testing.
    - *Environment.* The testing is done in a non-suitable environment size that does not provide accurate results.
5. ***Awareness of the importance***. The lack of understanding on the importance of non-functional testing from business and developers. They have the idea where if the system does what it is supposed to do, then the system is fine.
6. ***Culture of the company***. Businesses and developers need to create the habit of considering non-functional testing for a system. Agile developers specifically should always take both functional and non-functional testing into consideration in the development stage.
7. ***Experience of the members***. Due to bad past experiences, the senior team members do provide extra attention to non-functional testing. However, the younger team members are more concerned on the functional aspects of the system. Other than that, the testing of non-functional requirements requires a set of skill or expertise to ensure proper testing is conducted
8. ***Lack of communication with customer***. Lack of communication with customer or customer representative leads to incorrect prioritization of requirements by the developers who may lack in understanding about the market.
9. ***Minimal documentation of the process***. There are no templates or past documentation to assist in new projects.
10. ***Incorrect individual performing the tests***. In an agile project, where the pace is fast, many tests are conducted by the developer themselves or either someone in the same department. This can cause an issue because similar

mindset testing the system is highly likely to get the same result. Due to this, a different member of the team, or better still, a tester should review or conduct the testing phase.

11. ***Over-reliance on manual testing***. There are many tools available to assist in the testing process but due to insufficient knowledge, the process is highly relied on manual testing.
12. ***Lack of training invested for team members***. Team members lack the skill to identify requirements or perform the testing to achieve desired results.
13. ***Lack of team effort***. Everyone has a task provided to complete and their focus will be individual and not realize by working in a team, knowledge can be shared and work can be done simultaneously. For example, testing the system as development is moving to fix errors on-the-go.

Table 5. 1 Factors Influencing Non-Functional Testing

Variable	Factors
Factors influencing Non-Functional Testing	Time constraint.
	Budget constraint.
	Failing to prioritize in the initial stage.
	Technical issues.
	Awareness of the importance.
	Culture of the company.
	Experience of the members.
	Lack of communication with customer.
	Minimal documentation of the process.
	Incorrect individual performing the tests.
	Over-reliance on manual testing.
	Lack of training invested for team members.
	Lack of team effort.

Table 5.2 describes the challenges faced by agile team members when conducting non-functional testing. All the challenges were identified from this study. The challenges are; Programmers Writes Tests, Lack of documentation throughout the process, Lack of testing skillset between team members to conduct proper testing, Lack of individuals assigned to conduct the testing, Requirements are too subjective, Infrastructure overhead.

Table 5. 2 Challenges Faced When Conducting Non-Functional Testing

Variable	Challenges
Challenges Faced When Conducting Non-Functional Testing	Programmers Writes Tests.
	Lack of documentation throughout the process.
	Lack of testing skillset between team members to conduct proper testing.
	Lack of individuals assigned to conduct the testing.
	Requirements are too subjective.
	Infrastructure overhead.

Lastly, Table 5.3 determines the practices agile team members can use as a guide to ensure that non-functional testing can be conducted smoothly in their projects. A total of 3 were new practices discovered in this study; Test planning should be done in the early stage of development, Understanding the proper usage of testing tools to assist with the testing process, Repeat tests multiple times to ensure consistent results.

Table 5. 3 Practices to Conduct Better Non-Functional Testing

Variable	Practices
Practices to Adopt for Conducting Better Non-Functional Testing	Non-functional testing requires clear requirements elicitation.
	Non-functional testing requires additional requirements documentation to help the process and other members.
	NFR and testing need to be reviewed by at least one member from different role.
	Understanding of the importance of non-functional testing is required.
	Test planning should be done in the early stage of development.
	Understanding the proper usage of testing tools to assist with the testing process.
	Repeat tests multiple times to ensure consistent results.

The information should provide sufficient knowledge and understanding on the significance of NFR testing, to know what challenges to expect when conducting non-functional testing and the methods to overcome obstacles that agile team members may face during the course of the testing.

5.5 Accomplishment of Research Objectives

The main goal of this research was to identify the factors and challenges in conducting NFR testing in ASD. We conducted a review on previous studies to identify the factors and to discover the challenges faced to conduct the testing of NFR in ASD. The findings will help agile team members to know what challenges to expect when conducting NFR testing in their project.

The second objective of this study was to validate the identified factors of conducting NFR in ASD. The factors obtained were included in the questionnaire to be validated and at the same time, new factors were discovered from the results of the quantitative study. After conducting several statistical tests via SPSS, all 13 factors identified were included in the final findings.

The third objective was to propose practices that can be adopted by agile team member when conducting non-functional testing. A total of 7 practices were determined from the results of the questionnaire and approved acceptable after conducting statistical tests via SPSS. These practices can act as a guide for agile team members to assist them on how to conduct non-functional testing in their projects.

5.6 Implications

Theoretical implication. The findings of this study determined the factors influencing non-functional testing in ASD namely *Time constraint*, *Budget constraint*, *Failing to prioritize in the initial stage, Technical issues, Awareness of the importance, Culture of the company, Experience of the members, Lack of communication with customer, Minimal documentation of the process, Incorrect individual performing the tests, Over-reliance on manual testing, Lack of training invested for team members* and *Lack of team effort*. Other than that, challenges faced by agile team members when conducting non-functional testing and practiced that can be adopted by agile team member to conduct non-functional testing in their projects were also identified. This study has provided findings that will serve as a guide to agile team members in understanding and implementing non-functional testing in ASD projects. We have presented the findings from the results of the quantitative study.

Practical Implication. The process of non-functional testing is known to be a long process which require a lot of work and for it to be conducted on an agile environment is definitely a challenge. However, if the agile team members could follow the best practices and have an early overview of the challenges that they may encounter, it is highly likely that non-functional testing can be included and conducted successfully in their projects.

5.7 Limitations

Although the findings were made for any ASD projects around the world, the respondents were mainly based in Malaysia. Moreover, we could not manage to evaluate the items with actual ASD practitioners as several respondents were practitioners from different software development methodologies and several were students with minimal work experience in the field. Instead, a write up on ASD and non-functional testing were provided to the respondents in order to guide them in having a clear idea of the topic.

5.8 Recommendations for Future Work

Future research should consider getting respondents specifically from an ASD team and conducting the study across various regions. By conducting the study across various regions, we will be able to gather more information and practices that are being used abroad. Besides that, instead of quantitative study like the one conducted in this study, a qualitative study such as interviews or focus groups can be conducted to identify more factors. This will allow respondents to be more expressive and detailed in their explanations.

5.9 Conclusion

ASD methodology is highly adopted by various companies due to the ability to complete projects in a short amount of time. However, due to various factors, non-functional testing is known to be left out by agile team members. Our aim was to identify influencing factors of conducting NFR testing in ASD. We reviewed previous studies and gathered existing factors, challenges and practices to be used as a bassline of our quantitative study's questionnaire. Other than the existing information, there were more factors, challenges and practices identified from the data gathered from the questionnaire and run through several statistical test via SPSS. The final findings would provide a better insight to the significance of conducting non-functional testing in agile projects. The findings will better prepare agile team members of the challenges that they might encounter when conducting non-functional testing and provide them with a guide of practices to ensure proper non-functional testing can be conducting in ASD projects. In a nutshell, the findings should serve as a guide to conduct non-functional testing in ASD environment.

References

Agile Manifesto. (2014). Retrieved from http://agilemanifesto.org

Akbayrak, B. (2000). A comparison of two data collecting methods:Interviews and questionnaires. Hacettepe Üniversitesi Egitim Fakültesi Dergisi, 18, 1-10.

Alexander, T. (2018). How to Adopt an Agile Testing Methodology. Retrieved from http://smartbear.com/products/qa-tools/what-is-agile-testing

Ambler, S. W. (2008). Beyond functional requirements on agile projects. Dr. Dobb's Journal, vol. 33–10, 64–66 p. Retrieved from https://www.researchgate.net/publication/252064264_Beyond_functional_requirements_on_agile_projects

Antón, A. (1997). Goal Identification and Refinement in the Specification of Information Systems. PhD Thesis, Georgia Institute of Technology. Retrieved from https://dl.acm.org/citation.cfm?id=269264

Bahiya, M. A., & Abdelhamid, M. (2015). Enhancement Approach for NFR Analysis in Agile Environment. International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering. Retrieved from https://ieeexplore.ieee.org/document/7381407

Bose, S., Kurhekar, M., & Joydip, G. (2014). Agile methodology in Requirements Engineering. SETLabs Briefings Online, February. Retrieved from https://www.semanticscholar.org/paper/An-Empirical-Study-on-the-Requirements-Engineering-Kassab/85ed0087b901dbc0d4171ead29f56653389cc1e0

Collins, E., & Lucena, V. (2010). Iterative Software Testing Process for Scrum and Waterfall Projects with Open Source Testing Tools Experience. In Proceedings of the 22nd IFIP International Conference on Testing Software and Systems (ICTSS'10). CRIM, 2010. 115-120 p. [ISBN: 978-2-89522-136-4] .Retrieved from https://www.researchgate.net/publication/324438959_Iterative_Software_Testing_Process_for_Scrum_and_Waterfall_Projects_with_Open_Source_Testing_Tools_Experience

Crispin, L., & Gregory, J. (2009). Agile testing: A practical guide for testers and agile teams. Pearson Education. Retrieved from http://index-of.co.uk/Software-Testing/AGILE_TESTING_-_A_PRACTICAL_GUIDE_FOR_TESTERS_AND_AGILE_TEAMS.pdf

Crispin, L., & Gregory, J. (2014). More Agile Testing: Learning Journeys for the Whole Team. Pearson Education. Retrieved from http://ptgmedia.pearsoncmg.com/images/9780321967053/samplepages/9780321967053.pdf

Cristina, R. C., Sabrina, M., & Daniela, S. C. (2016). Agile Team Members Perceptions on Non-functional Testing: Influencing Factors from an Empirical Study. 11th International Conference on Availability, Reliability and Security. Retrieved from https://ieeexplore.ieee.org/abstract/document/7784622

Davies, D., & Doodd, J. (2002). Qualitative research and question of riger. Qualitative Health research, 12(2), 279-289. Retrieved from http://www.dspace.up.ac.za/bitstream/handle/2263/28048/02chapter3.pdf?sequence=3

Davis, A. (1993). Software Requirements: Objects, Functions and States. Prentice Hall. Retrieved from https://dl.acm.org/citation.cfm?id=113586

DeCoster, J. (1998). Overview of factor analysis. Retrieved from http://stat-help.com/factor.pdf

Denisse, M. (2018). Traditional vs. Agile Software Development Method: Which One is Right for Your Project? Retrieved from https://dzone.com/articles/traditional-vs-agile-software-development-method-w

Dybå, T., & Dingsøyr, T. (2009). What Do We Know about Agile Software Development? Software, IEEE. 26. 6 - 9. 10.1109/MS.2009.145. Retrieved from https://ieeexplore.ieee.org/document/5222784

Edward, K., & Susannah, F. (1995). Software Testing in the Real World: Improving the Process. Addison-Wesley, Reading, MA, USA. Retrieved from https://dl.acm.org/citation.cfm?id=217720

Eliane, F. C. (2012). Software test automation practices in agile development environment: An industry experience report. In: Proceedings of the 7th International Workshop on Automation of Software Test, pp. 57–63. Retrieved from https://dl.acm.org/citation.cfm?id=2663620

Faisandier, A. (2012). Systems opportunities and requirements, vol. 2. Engineering and Architecting Multidisciplinary Systems. Sinergy'Com, France.

Glenn, V. (2005). A Simple Model of Agile Software Practices - or - Extreme Programming Annealed, in Object-Oriented Programming, Systems, Languages, and Applications, New York, pp. 539-545. Retrieved from https://dl.acm.org/citation.cfm?id=1094854

Glinz, M. (2005). Rethinking the notion of NFR. Proceedings of the Third World Congress for Software Quality, 55–64. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.5284&rep=rep1&type=pdf

Hanan, H., Rocky, S., Jianwei, N., & Travis, D. (2017). Rethinking Security Requirements in RE Research. Retrieved from https://pdfs.semanticscholar.org/8aae/a45a11a90b6bb4158ae61a509258bcf5688b.pdf

Huang, L., & Boehm, B. (2006). How Much Software Quality Investment Is Enough: A Value-Based Approach. IEEE Software, Vol. 23(5), pp. 88-95, doi: 10.1109/MS.2006.127. Retrieved from https://dl.acm.org/citation.cfm?id=1159149

Hutcheson G., & Sofroniou N. (1999). The multivariate social scientist: introductory statistics using generalized linear models. London: Sage Publication.

IEEE. Guide to the Software Engineering Body of Knowledge - IEEE SWEBOK, 2004. Retrieved from: https://pdfs.semanticscholar.org/9151/665c0b7f49aaf260d18d4177c685638a0b8e.pdf

IEEE. (1990). Standard Glossary of Software Engineering Terminology. IEEE Standard 610.12-1990. Retrieved from https://ieeexplore.ieee.org/document/159342

Itti, H., & Rajender, S. C. (2015). Software Test Process, Testing Types and Techniques. International Journal of Computer Applications (0975 – 8887) Volume 111 – No 13, February. Retrieved from https://pdfs.semanticscholar.org/0fbe/1b5515e747025d950658fbc039e98b29b801.pdf

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The Unified Software Development Process. Retrieved from https://dl.acm.org/citation.cfm?id=309683

Jai, N. G., & Mehtre, B. M. (2015). Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050915019870

Kaiser, H. F. (1974). An Index of Factorial Simplicity. PSYCHOMETRIK, 39(1), 31–36. Retrieved from https://pdfs.semanticscholar.org/f4f5/31ba422264ec4ff7fc09db8680f8299fc706.pdf

Karuturi, S., & Malle, G. M. (2017). Research on Software Testing Techniques and Software Automation Testing Tools. International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). Retrieved from https://ieeexplore.ieee.org/document/8389562

Kiran, K., & Arvind, K., (2013). Impact of NFR on Requirements Evolution. Sixth International Conference on Emerging Trends in Engineering and Technology. Retrieved from https://dl.acm.org/citation.cfm?id=2606288

Klein, P. G. (2008). The Make-or-Buy Decisions: Lessons from Empirical Studies. Handbook of New Institutional Economics, pp 435-464. Retrived from https://pdfs.semanticscholar.org/2253/f1183e3ec869a5f839d6e6d6fdfa2e6c4b98.pdf?_ga=2.16296310.1941105328.1568642655-1969972726.1567949410

Kothari, C. R. (2004). Research Methology: Methods and Techniques (2nd ed.). India: New Age International.

Kotonya, G., & Sommerville, I. (1998). Requirements Engineering: Processes and Techniques. Retrieved from https://dl.acm.org/citation.cfm?id=552009

Lawrence, B., Wiegers, K. & Ebert, C. (2001). The top risk of requirements engineering. Software, IEEE, vol. 18–6, 62–63 p. Retrieved from https://ieeexplore.ieee.org/abstract/document/965804

Leedy, P. D. (1993). Practical research: planning and design. New Jersey: Prentice-Hall.

Mabrok, M. A., Efatmaneshnik, M., & Ryan, M. J. (2015). Including NFR in the Axiomatic Design Process. IEEE Systems Journal, 1–11. Retrieved from https://www.researchgate.net/publication/282331926_Including_Non-Functional_Requirements_in_the_Axiomatic_Design_Process

Marizia D. et. Al. (2010). An Investigation into the Notion of NFR. SAC'10, Sierre, Switzerland. Retrieved from https://www.researchgate.net/publication/261073604_On_non-functional_requirements_A_survey

Martens, N. (2011). The impact of NFR on project success. Retrieved from http://www.cs.uu.nl/education/scripties/pdf.php?SID=INF/SCR-2010-038

Martin, G. (2007). On NFR. 15th IEEE International Requirements Engineering Conference, pages 21–26, 15-19 Oct. Retrieved from https://ieeexplore.ieee.org/document/4384163

Meier, J., Farre, C., Bansode, P., Barber, S., & Rea, D. (2007). Performance testing guidance for web applications: patterns & practices. Microsoft press. Retrieved from https://dl.acm.org/citation.cfm?id=1461439

Mohammad, K. (2014). An Empirical Study on the Requirements Engineering Practices for Agile Software Development. 40th Euromicro Conference on Software Engineering and Advanced Applications. Retrieved from https://ieeexplore.ieee.org/abstract/document/6928819

Mylopoulos, J., Chung, L., & Nixon, B. (1992). Representing and Using Nonfunctional Requirements: A Process- Oriented Approach. IEEE Transactions on Software Engineering 18, 6 (June). 483-497. Retrieved from https://ieeexplore.ieee.org/document/142871

Ncube, C. (2000). A Requirements Engineering Method for COTS-Based Systems Development. PhD Thesis, City University London. Retrieved from https://ieeexplore.ieee.org/document/142871

Nerur, S., & Balijepally, V. (2007). Theoretical Reflections on Agile Development Methodologies. Comm. ACM, vol. 50, no. 3, pp.79–83. Retrieved from https://dl.acm.org/citation.cfm?id=1226739

Neuman, W. L. (2006) Social Research Methods: Qualitative and Quantitative Approaches 6th Edition, Pearson International Edition, USA.

Pamela, Z. (1997). Classification of research efforts in requirements engineering. ACM Comput. Surv., 29(4):315–321. https://dl.acm.org/citation.cfm?id=267581

Phellas, C., Bloch, A. & Seale. (2011). Structured methods: Interviews, questionnaires and observation Researching Society and Culture. 3 ed. London.

Professional QA. (2018). Retrieved from http://www.professionalqa.com/traditional-testing-vs-agile-testing

Raimundas, M. (2005). Process Support for Requirements Engineering A Requirements Engineering Tool Evaluation Approach. PhD thesis, NTNU. Doctoral theses at NTNU:142. Retrieved from https://www.semanticscholar.org/paper/Process-Support-for-Requirements-Engineering%3A-A-Matulevicius/115d8fc79b12e840da36ea7b57e29fc025937c3f

Rijwan, K., Akhilesh, K. S. & Dilkeshwar, P. (2016). Agile Approach for Software Testing Process. Proceedings of the SMART -2016, IEEE Conference ID: 39669. 5th International Conference on System Modeling & Advancement in Research Trends, 25th_27'h November. Retrieved from https://ieeexplore.ieee.org/document/7894479

Robertson, S., & Robertson, J. (1999). Mastering the Requirements Process. ACM Press. Retrieved from https://dl.acm.org/citation.cfm?id=312381

Sean, I. (2001). Challenges of Requirements Elicitation. Louis University of Missouri–St. Louis. [Online document]. Retrieved from https://www.umsl.edu/~sauterv/analysis/Fall2010Papers/Isserman/

Siron, R., Tasripan, M. A. & Majid, M. Y. (2013). A Study of Quality of Working Life amongst Managers in Malaysian Industrial Companies. Journal of Business and Economics, ISSN 2155-7950, USA, Volume 4, No. 7, pp. 561-570. Retrieved from http://www.academicstar.us/UploadFile/Picture/2014-6/201461495752684.pdf

Sommerville, I. (2010). Software engineering — 9th ed. Retrieved from https://dl.acm.org/citation.cfm?id=1841764

Standish Group. (2014). The Chaos Report. Retrieved from https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

Tavakol, M. & Dennick, R. (2011). Making sense of Cronbach's alpha. International Journal of Medical Education. Retrieved from https://www.ijme.net/archive/2/cronbachs-alpha.pdf

Verndon, D., & McGraw, G. (2004). Risk Analysis in software Design. IEEE Security and Privacy.2,4.32-37 (July/August 2004). Retrieved from https://ieeexplore.ieee.org/document/1324606

Vikas, B., & Ravi, P. G. (2012). On NFR: A Survey. IEEE Students' Conference on Electrical, Electronics and Computer Science. Retrieved from https://ieeexplore.ieee.org/abstract/document/6184810

Ville, T. H., Casper, L., Daniela, D., & Maria, P. (2015). Mapping Study on Requirements Engineering in Agile Software Development. 41st Euromicro Conference on Software Engineering and Advanced Applications. Retrieved from https://ieeexplore.ieee.org/document/7302452

Warren, C. A. (2011). The Need for Functional Security Testing. Retrieved from https://pdfs.semanticscholar.org/f0ab/68c7d4be4adb0adc19bb5879b9772c3f200d.pdf

Westfall, L. (2005). Software requirements engineering: what, why, who, when, and how. Software Quality Professional 7.4: 17. Retrieved from https://www.researchgate.net/publication/293118075_What_Why_Who_When_and_How_of_Software_Requirements

Wiegers, K. (2003). Software Requirements, 2nd edition. Microsoft Press. Retrieved from https://dl.acm.org/citation.cfm?id=862054

APPENDICES

APPENDIX A: QUESTIONNAIRE

Part 1: Demographic Questions

1. In what year were you born?

   ☐

2. State your gender.
   - ☐ Male
   - ☐ Female
3. Highest level of education?
   - ☐ Tertiary (colleges & universities)
   - ☐ Secondary
   - ☐ Primary
4. Have you heard about agile software development?
   - ☐ Yes
   - ☐ No
5. Have you been directly involved in software development process?
   - ☐ Yes
   - ☐ No
6. Have you heard about software testing?
   - ☐ Yes
   - ☐ No
7. Have you been involved in testing NFR in projects?
   - ☐ Yes
   - ☐ No
8. State the number of years of work experience you have acquired. (Involvement in Software Development)
   - ☐ Less than 5 years
   - ☐ 5 to 10 years
   - ☐ 11 to 15 years
   - ☐ More than 15 years
   - ☐ None

Part 2: Questionnaire

		Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1.	I understand there are various types of software development methodologies available.					
2.	In software development process, agile methodology is widely used in projects.					
3.	I prefer agile methodology compared to the traditional methodology.					
4.	It is important to conduct software testing on every project.					
5.	Non-functional testing is more important than Functional testing.					
6.	Non-functional testing is required in all projects.					
7.	If provided a choice, I would choose to conduct non-functional testing in all projects.					
8.	Non-functional testing process is started when the project begins.					
9.	Non-functional testing process should start as soon as a project begins.					
10.	Non-functional testing is done by anyone in the team.					
11.	Non-functional testing should only be done by a specialist.					
12.	Non-functional testing is not taken seriously by testers.					
	Below are the known to be the influencing factors of non-functional testing.					
13.	Time constraint.					
14.	Budget constraint.					
15.	Failing to prioritize in the initial stage.					
16.	Technical issues.					
17.	Awareness of the importance.					

18.	Culture of the company.					
19.	Experience of the members.					
20.	Lack of communication with customer.					
21.	Minimal documentation of the process.					
22.	I face various challenges in conducting non-functional testing.					
23.	I am able to overcome the challenges faced.					
24.	Challenges faced in every project are similar.					
25.	Non-functional testing requires clear requirements elicitation.					
26.	Non-functional testing requires additional requirements documentation to help the process and other members.					
27.	NFR and testing need to be reviewed by at least one member from different role.					
28.	Understanding of the significance of non-functional testing is required.					

29. In your opinion, what are the factors that influence non-functional testing in agile environment?
(Enter the factors followed by a number between 1-5 indicating how influential the factor is. 5 being most influential) Example: Not equipped with skill set = 4; Lack of team effort = 5;

30. What were the challenges faced when conducting non-functional testing?
(Enter the challenges followed by a number between 1-5 indicating how challenging it is. 5 being most challenging) Example: Conducting testing with no prior training = 4; Lack of manpower to perform complete testing = 5;

31. How did your team you/your team overcome the challenges stated above?
(Enter the practices followed by a number between 1-5 indicating how effective the practice is. 5 being most effective) Example: Learned the right tools to assist with the testing = 4; Created a guideline to assist with the testing = 5;

APPENDIX B: EXPERT REVIEW

		Expert 1	Expert 2	Expert 3	Expert 4	Revised
Agile Software Development Methodology						
1.	I understand there are various types of software development methodologies available.					I understand there are various types of software development methodologies available.
2.	In software development process, agile methodology is widely used in projects.					In a software development process, agile methodology is widely used in projects.
3.	I prefer agile methodology compared to the traditional methodology.			*Looks not related*		I prefer agile methodology compared to the traditional methodology.
Software Testing (Non-Functional Testing)						
4	It is important to conduct software testing on every project.					It is important to conduct software testing on every project.
5.	Non-functional testing is more important than Functional testing.					Non-functional testing is more important than Functional testing.
6.	Non-functional testing is not taken seriously by testers.					Non-functional testing is not taken seriously by testers.
7.	Non-functional testing is required in all projects.					Non-functional testing is required in all projects.
8.	If provided a choice, I would choose to conduct non-functional testing in all projects.					If provided a choice, I would choose to conduct non-functional testing in all projects.

Non-Functional Testing Process						
9.	Non-functional testing process is started when the project begins.					Non-functional testing process is started when the project begins.
10	Non-functional testing process should start as soon as a project begins.					Non-functional testing process should start as soon as a project begins.
11.	Non-functional testing can be done by anyone in the team.	*Revise*	*Revise*			Non-functional testing is done by anyone in the team.
12.	Non-functional testing should only be done by a specialist.					Non-functional testing should only be done by a specialist.
Factors influencing Non-Functional Testing						
13.	Time constraint.					Time constraint.
14.	Budget constraint.					Budget constraint.
15.	Failing to prioritize in the initial stage.					Failing to prioritize in the initial stage.
16.	Technical issues.					Technical issues.
17.	Awareness of the importance.					Awareness of the importance.
18.	Culture of the company.					Culture of the company.
19.	Experience of the members.					Experience of the members.
20.	Lack of communication with customer.					Lack of communication with customer.
21.	Minimal documentation of the process.					Minimal documentation of the process.
22.	In your opinion, what are the factors this?	*Define 'this'*		*Define 'this'*		In your opinion, what are the factors that influence non-functional testing in agile environment?

Challenges Faced When Conducting Non-Functional Testing						
23.	I face various challenges in conducting non-functional testing.					I face various challenges in conducting non-functional testing.
24.	I am able to overcome the challenges faced.					I am able to overcome the challenges faced.
25.	What were the challenges faced when conducting non-functional testing and how did you/your team overcome them?	*Split into 2 questions*	*Split into 2 questions*		*Split into 2 questions*	What were the challenges faced when conducting non-functional testing?
26.	Challenges faced in every project are similar.					Challenges faced in every project are similar.
Practices to Adopt for Conducting Better Non-Functional Testing						
27.	Non-functional testing requires clear requirements elicitation.					Non-functional testing requires clear requirements elicitation.
28.	Non-functional testing requires additional requirements documentation to help the process and other members.					Non-functional testing requires additional requirements documentation to help the process and other members.
29.	NFR and testing need to be reviewed by at least one member from different role.					NFR and testing need to be reviewed by at least one member from different role.
30.	Understanding of the importance of non-functional testing is required.					Understanding of the importance of non-functional testing is required.
31.	What were the challenges faced when conducting non-functional testing and how did you/your team overcome them?	*Split into 2 questions*	*Split into 2 questions*		*Split into 2 questions*	How did your team you/your team overcome the challenges faced when conducting non-functional testing?