Investigating the Factors that Influence the Misalignment between Developers and Testers in Agile Organizations

A Masters Dissertation presented by Unathi Mbekela (MBKUNA002) to the

Department of Information Systems

University of Cape Town

In partial fulfilment of the requirements for INF5004W

Allocated Supervisor: Prof. Irwin Brown

15 August 2016

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the APA convention for citation and referencing. Each contribution to, and quotation in, this literature review for the research entitled ***'Investigating the Factors that Influence the Misalignment between Developers and Testers in Agile Organizations'*** from the work(s) of other people has been attributed, and has been cited and referenced.
3. This paper is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
5. I acknowledge that copying someone else's assignment, essay or paper, or part of it, is wrong, and declare that this is my own work.

Unathi Mbekela Signed by candidate

Signature Removed

Date: 15/08/2017

Abstract

The concept of alignment has been addressed in the context of various divisions within organizations but very little research investigates the alignment of the roles within specific sub-units in an organization. Research shows evidence of a misalignment between the role of the software tester and the software developer in software development teams specifically in organizations that adopt agile methodologies to manage their software development projects. It is this misalignment between these two roles and the lack of research on the factors that influence this phenomenon that prompted the study. The study aims to investigate the factors which influence misalignment between developers and testers in agile organizations with specific focus on the social dimension of alignment contrary to most studies that merely address the intellectual dimension of alignment.

The research methodology followed a positivist, quantitative and deductive approach. An online questionnaire was designed and distributed to respondents in South Africa (SA) and United States of America (USA). The results show that there are four factors that have an overall influence on the misalignment between developers and testers in agile software development teams. These factors are (1) process non-compliance combined with lack of accountability, (2) conflicting interpersonal skills, (3) lack of shared domain knowledge, specifically lack of developers' knowledge about testing and (4) poor collaboration. Future research can proceed to identify the strategies that agile organizations can adopt alleviate this problem of misalignment.

Table of Contents

List of Tables

Table of Figures

1. CHAPTER ONE: INTRODUCTION

1.1 Background

Agile methodologies evolved from the agile manifesto which was launched by a group of industry specialists in 2001 (Beck, et al., 2001). It emphasizes people and relationships over procedures and tools because the success of a team is often not determined by the procedures and tools that are used but more on the people that are involved and the type of working relationship they have within the team. In agile organizations, it is important to have people skills because no matter how excellent individuals may be at their job, if they are not team players and cannot work well with others, this may be detrimental to the overall work done in the team (Beck, et al., 2001). The use of agile methodologies to manage software development projects has become more popular because this method of software development allows organizations to react rapidly to any changes and ensures that teams work efficiently by maximizing all their resources while minimizing the cost. The implementation of agile methodologies suggests that teams employ a dedicated tester/s in each software development team because testing is a crucial task in the software development process (Borland, 2012). This means that testing will not only be done by the developers while they are writing the code but it will also be done by the software tester once development is completed (Kettunen, Kasurinen, Taipale, & Smolander, 2010). Therefore, developers and testers need to learn to work interdependently and communicate effectively to achieve this task (Linders, 2013). Lagastee (2013) and Zhang et al. (2014) reveal that there exists some form of conflict in the relationship between developers and testers and this conflict results in a breakdown in communication and collaboration between the two parties. The phenomenon appears to be more apparent in agile organizations than traditional plan-driven organizations hence the interest in exploring this problem within this context.

1.2 Problem Statement

The relationship between developers and testers in software (SW) development organizations is a subject of concern because studies have revealed that there is poor collaboration between the two parties (Dhaliwal, Onita, Poston, & Zhang, 2011). Moreover, developers and testers have differing roles even though they are expected to work closely within their team (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014). It is important for developers and testers to have a good working relationship because the work they do is critical to the success of the overall organization especially an agile organization. It is important to ensure that there is good collaboration and communication between these two parties (Cohn, Leffingwell, Larman, & Vodde, 2013).

Davis (2014) suggests that managers of SW development teams should emphasize the importance of teamwork, having the right approach towards team members and that every member of the SW development team plays an important role towards the quality of the end product. Furthermore, Davis (2014) adds that the managers should initiate and oversee discussions between developers and testers prior to starting projects so that both parties have an opportunity to discuss the work and the approach to follow in doing the work so that there is a shared understanding and common goal in the team and subsequently to ensure that there is a sense of unity and collaboration between the team members.

The research investigates the factors that influence misalignment between developers and testers in agile organizations. Prior research highlights that there is a misalignment/lack of alignment between developers and testers but does not extensively identify nor address the factors that influence this problem (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014).

1.3 Research Questions

The main research question is:

- *What factors influence the misalignment between developers and testers in agile organizations?*

1.4 Structure of the Study

The study commences with a literature review discussing an analogy between traditional plan-driven organizations and agile organizations, followed by a definition of misalignment in the context of this study, followed by a discussion of the agile software team and a discussion of the factors that influence misalignment. A three-layer conflict model depicts the factors that influence misalignment between developers and testers. This section is followed by the research methodology which discusses the research philosophy and paradigm, research approach, sample strategy, data collection and analysis. Following this section is the statistical analysis of the data as well as a discussion of the findings. The last section discusses the research limitations, conclusion and recommendations for future research.

2. CHAPTER TWO: LITERATURE REVIEW

2.1 Traditional Plan-driven Organizations and Agile Organizations

Traditional plan-driven organizations are governed by processes which set out how various projects should be managed from start to finish. Traditional-plan driven organizations are more concerned about mastering a process with extensive planning for a software development project well in advance without taking into consideration that there may be unforeseen changes during the project lifecycle. These types of organizations do not plan for change but instead try and control change through the process of gathering requirements, analyzing the requirements and subsequently designing and developing these requirements (McElfish, 2011).

On the contrary, the agile methodology is not a process but a set of principles which underpin the software development process. Agile methods involve developing software rapidly in an environment where requirements are constantly changing and the release of software to customers occurs on a regular basis. Agile methods are very fluid in nature because they allow for teams to identify various ways of performing tasks to improve how software is developed. Working software is the primary emphasis of agile methods (Greer & Hamon, 2011).

2.2 Alignment and Misalignment

Since the official launch of the agile manifesto in 2001, the word 'agile' has become quite popular in many organizations and the adoption of agile methodologies has become very prominent especially in IT organizations (Gandomani, Zulzalil, Ghani, & Sultan, 2012). Since the research will also be focusing on the misalignment between developers and testers in agile organizations, there is a need to define what misalignment means in the context of this research because some authors refer to alignment as a fit, a link, a bridge, a state of harmony or an integration (Aversano, Grasso, & Tortorella, 2012).

Onita & Dhaliwal (2011) define alignment as the strategic operational fit between an organization's development functions and testing functions which they refer to as development-testing alignment. This definition emphasises the fit between development and testing functions, but it only focuses on the strategic and operational aspects between the two parties and does not consider social aspects such as communication, collaboration and team work.

Dhaliwal, Onita, Poston, & Zhang (2011) also make reference to the alignment between developer and tester subunits within the IT organization but they do not explicitly define alignment. They do however address social factors such as shared understanding, partnerships and competencies between developers and testers. Due to the fact they do not provide a useful definition of alignment, this study further investigates a suitable definition of alignment that will be relevant.

Aversano, Grasso, & Tortorella (2012) define alignment as the process whereby the organizational procedures, objectives and activities of the overall organization are aligned to the information systems processes that they depend on, and is akin to what has been described as the intellectual dimension of alignment by Reich & Benbasat (2000).

Hanson, Melnyk, & Calantone (2011) further define alignment as a condition that can be created, one that directly influences competitive advantage and encourages teamwork and harmony within the organization. This type of alignment encourages understanding of the project strategy, acceptance of the project strategy, having a link between combined efforts in the team and setting standards/targets that need to be met by the team. This definition of alignment refers to social issues much like the social dimension alignment of Reich & Benbasat (2000).

To define misalignment using Hanson, Melnyk, & Calantone (2011) it can be inferred that misalignment is the lack of teamwork and harmony within the organization. This latter definition is thus applied to the study in the context of developers and testers because the aim of the research is to investigate the factors that influence misalignment between the two parties. Furthermore, the social dimension is justified in this definition because the agile manifesto emphasizes people and relationships over procedures and tools (Beck, et al., 2001).

2.3 Agile Software Development Team

Contrary to traditional plan-driven organizations which emphasize a command-and control management approach, the agile management approach encourages self-managing teams which allows all the team members to collectively manage and carry out the team tasks (Moe, Dingsøyr, & Dybå, 2010). The agile software development team is made up of several individuals who play a significant role in the overall success of a project.

- Developers - SW developers are usually involved in the SW development lifecycle (SDLC) from the planning until the maintenance phase; because they initially meet with the customers or end users of the SW they intend to develop. In these meetings, they discuss the requirements of the customer and whether these requirements are feasible. Thereafter all the requirements are documented and forwarded to the programmers who in turn write the code. The code is then tested by SW testers and if any bugs or issues arise, these are sent back to the programmers. This is done until the SW developer is satisfied with the SW, which they then demonstrate to the customer. If the customer is satisfied, the product is officially released (Yu, Wooi, Wai, & Soo, 2012).

- Testers - SW testers evaluate the quality of the SW that has been developed by SW developers to ensure that it meets the requirements as specified in the requirements specification document. SW testers usually join the SDLC at the implementation stage, once all the development has taken place and the SW is ready to be tested. SW testers are responsible for ensuring the overall quality of the SW has been met, and prevent SW to be released if it is not free of bugs, errors and other problems (Yu, Wooi, Wai, & Soo, 2012). Testers in agile organizations are expected to perform a number tests to evaluate the quality of software prior to the software being released to go live and be utilized by external stakeholders and clients.

2.3.1 Software Testing

The aim of software testing is to evaluate the quality of a software program against defined expectations and documented standards (Lewis, 2016) and further provide valuable information for developers. Contrary to the general view, software testing in not merely about "finding bugs" but ensuring that software is of an acceptable standard and quality.

To evaluate the quality of a given software program, developers and testers must exhaust various types of tests and quality assurance checks. These tests and checks are either manual or automated or a combination of both (Lewis, 2016). Furthermore, the goal of testing is to detect and high severity discrepancies to the documented and expected results so that those issues can be resolved before the software is officially released (Parveen, Tilley, & Gonzalez, 2007).

Software testing can be performed before the official release of software and after the official release. Testing that is performed prior to release is referred to as alpha testing and is performed by developers and testers within the organization. While testing that is performed after official release is referred to as beta testing and is usually performed by the end-users or clients on the client site (Lewis, 2016). Moreover, Parveen, Tilley, & Gonzalez (2007) further add that the testing performed by developers is different to the testing performed by testers. Developers focus on white box testing which only involves testing of the code that they have developed and not the actual user interface. While testers focus on black box testing which does not look into the code but focuses on the actual front end and final product that will be released to the users. The type of testing performed by testers in agile organizations also includes pre-production testing, user acceptance testing, retesting after a bug or issue has been resolved and regression testing to ensure that any bug fixes do not introduce new issues to the software that was already fully functional and signed off.

It is evident that the role of the developer and tester are somewhat misaligned, because the developer aims to design and develop working SW efficiently while maximizing resources and minimizing the time used for the whole development process. Testers on the other hand strive to ensure that the SW produced does not only work, but that it is of acceptable quality and effectively does what it has been set out to do (Zhang et al., 2014).

Research shows that misalignment often arises between developers and testers in agile SW development teams. This misalignment arises because of the differing roles, objectives and skill sets of the developers and the testers. While the developer is responsible for designing the SW as specified in the requirements specification, developing the code for the SW and making sure that the SW does what it has been set out to do; the testers' responsibilities involve identifying flaws, bugs or problems in the SW that the developers have produced and overseeing the quality of the end product (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014). Furthermore, due to the differing types of testing performed by the developers and testers, conflict arises because developers claim that they tested the software thoroughly and did not find any deviations from the requirements. While testers identify problems in the software which in turn results in developers having to revisit their code and fix the issues.

2.4 Factors that Influence Misalignment between Developers and Testers

For the developer and the tester to collaborate and communicate effectively, they need to be placed in one SW development project team because the two parties need to work very closely to be successful in SW development projects (Cohn et al., 2013). Having a developer and a tester in one agile team has raised the argument of whether a tester is necessary or not in the team because developers do somewhat test their work before it is officially released to production environments (Sumrell, 2007).

The need was identified for testers to form part of an agile team because the testing that the developers do relates to the units of the code that they have developed and not on the overall finished product (Sumrell, 2007). Therefore, verifying that the code is correct does not mean that the overall functionality works according to the specification (Sumrell, 2007).

Zhang et al. (2014) propose a three-layer misalignment model which depicts the factors that influence the misalignment between developers and testers. The three-layer misalignment model is illustrated in Figure 1 below:

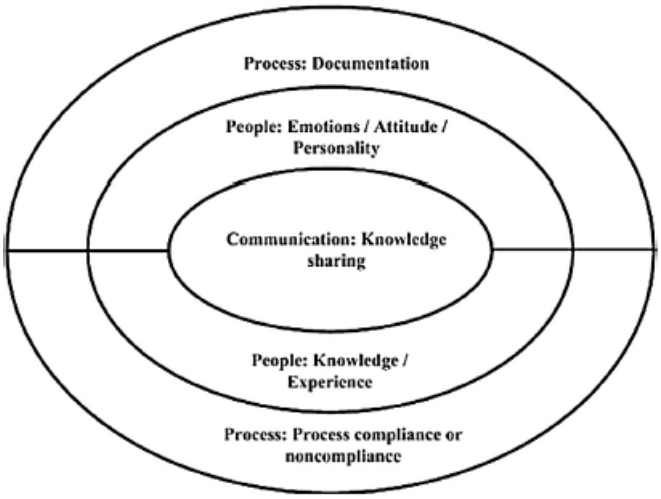

Figure 1 Three-Layer Misalignment Model

Zhang et al. (2014)

The model posits that the source of misalignment stems from three components: process, people and lack of communication, but centrally lack of communication.

Drawing from Figure 1 it can be inferred that the following factors contribute to the misalignment between developers and testers.

2.4.1 Lack of Communication

According to Littlejohn & Foss (2011) communication has various definitions which are context specific. Having one universal definition of communication would be an injustice to the term. Communication has been defined as the medium through which information is transmitted from one point to another (Sargunar, 2011). Another definition of communication is a situation where a source transfers a message to a receiver with the desired hope of influencing the receiver's actions (Littlejohn & Foss, 2011). Furthermore, Muller (2013) defines communication as the verbalized exchange of information and ideas between two or more parties.

The most relevant definition of communication which can be applied to the current study is the definition by Muller (2013). Thus, communication can be defined as the verbalized exchange of information and ideas between developers and testers in an agile organization. The lack of communication would mean that there is minimal or no exchange of information and ideas between developers and testers in agile organizations.

Lack of communication between developers and testers in project teams plays a big role in contributing to the misalignment between these two parties. Organizations should identify strategies that will create an environment where communication can be easy between these two parties (Zhang et al., 2014). Since both developers and testers play an important role in ensuring that the quality of the SW product is satisfactory, there ought to be a sense of togetherness and unity in their work ethic. Miscommunication between them has adverse effects on achieving the strategic objectives of the SW development unit (Ammann & Offutt, 2008).

2.4.2 Poor Collaboration

Collaboration is the process of directly interacting with other people in order to accomplish a specific task (Brown, Lindgaard, & Biddle, 2011). Contary to meetings, collaborative interactions are not lead by an agenda, they do not involve a single individual directing the meeting and there is initially no collective purpose for the collaboration. Collaboration encourages synergy among team members which in turn makes is easier for individuals to cooperate, share knowledge and ideas and also develop a level of understanding on the tasks that need to be carried out (Brown, Lindgaard, & Biddle, 2011).

Collaboration is required for tasks which prove to be more complex for an individual to perform. Hence agile organizations encourage collaboration because in a software development project, there are number of complex tasks that have to be carried out and these cannot be done by a single person. Software development projects thrive on team work, each individual is allocated to their area of expertise and they will collaborate with other team members to successfully complete a project (Helquist, Deokar, Meservy, & Kruse, 2011).

Brown, Lindgaard, & Biddle (2011) posit that agile organizations encourage collaboration more than traditional-plan driven organizations which means that developers and testers in an agile team cannot have poor collaboration regardless of the fact that their roles are different. Poor collaboration has a very negative impact on the success of any SD project.

Marczak & Damian (2011) emphasize the importance of collaboration and communication specifically in a SW development team to ensure the success of a project. Poor collaboration between those involved in performing the core tasks in the SW development team could result in a failure of the project. Hence poor collaboration between developers and testers contributes to misalignment (Ammann & Offutt, 2008).

2.4.3 Process Non-Compliance

The three-layer misalignment model of Zhang et al. (2014) suggests that developers are not efficient in updating bug tracking tools and in informing the testers of changes or updates to code, which means that developers struggle to comply with the processes set out for the team. This non-compliance with process further contributes to misalignment. For developers and testers to have a good working relationship and be effective as a team, they need to comply with the processes which underpin agile methodologies (Tripathi & Goyal, 2014).

Furthermore, lack of process design and documentation emphasizes knowing what the processes are before they can be managed effectively because it is not possible to manage what is not known (Kohlbacher & Gruenwald, 2011). The following factors need to be met before it can be concluded that there is a lack of process design and documentation: lack of process documentation, lack of use and update of process documentation and lack of definition of input and outputs for each process (Kohlbacher & Gruenwald, 2011). Lack of process performance management refers to the fact that companies emphasize the measurement of roles and functions in an organization but neglect the measurement of processes. If processes are not measured they will not be mastered. People and expertise also influence process non-compliance (Kohlbacher & Gruenwald, 2011). If developers and testers do not have sufficient knowledge on how to implement the process, then they may not fully comply with it (Tripathi & Goyal, 2014).

Furthermore, if developers and testers have limited problem solving, process improvement and decision making techniques, they may struggle to comply with processes (Kohlbacher & Gruenwald, 2011).

2.4.4 Lack of Shared Domain Knowledge (SDK)

According to Bruun & Stage (2012) developers have a completely different mindset to that of testers and thus they struggle to accept that there are bugs or flaws in the software that they have developed. They usually take it personally when testers identify bugs in their software because they view their software as part of themselves. This results in a misalignment between these two parties because developers are convinced that the work they have developed is according to the specification and will meet the needs of the users. Testers on the other hand have a broader knowledge about the software that needs to be developed because they also understand the overall business and the functional aspects of the software (Cohen, Birkin, Garfield, & Webb, 2004). Testers usually pay attention to detail and work as a team unlike developers who prefer to work individually without collaborating with other team members (Cohen, Birkin, Garfield, & Webb, 2004).

The role of the developer involves analyzing requirements, writing code that subsequently makes up the software. Developers are also expected to test their code for any obvious flaws or bugs which may cause the software not to function the way that it was intended (Yu, Wooi, Wai, & Soo, 2012). On the contrary, the role of a tester involves verifying and validating the software that has been produced by the developers. The focus of a tester is fault-finding and attempting to 'break' the software that was developed by the developers (Yu, Wooi, Wai, & Soo, 2012). Testers ensure that any anticipated problems are resolved before the software is made available to the end-users and that the quality of the software is unquestionable (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014).

Even though developers and testers need to work closely for the successful completion of a software development project, their roles are completely different and conflicting and thus their mind-sets are not the same. In order for agile software development teams to be successful and for the teams to achieve alignment, there needs to be knowledge sharing between developers and testers.

2.4.5 Lack of Accountability

Since SW testers do the final testing before a product is released to the production environment, they are often blamed when bugs or issues are found in the production environment and the developers are usually safe from the blame (Lagestee, 2013). Onita & Dhaliwal (2011) argue that SW development projects fail because of the lack of testing from both the developers and the testers and due to the lack of unity between these parties.

It is important for developers and testers to have mutual project goals so that they are no working against each other but with each other to ensure the success of a software development project. Trust plays a major role in a software development team therefore it is important that developers and testers become open with one another on the work they are doing, the timeframes for the work and whether it is successfully completed. This ensures that there is transparency between developers and testers and thus accountability (McHugh, Conboy, & Lang, 2012).

In a software development, agile team everyone is expected to estimate the amount of time it will take them to complete a specific task (Waters, 2007). This can be a development task, testing task or a product owner task. Once an individual has allocated the task to themselves, they are responsible to ensure that the task is completed in the amount of time that was specified. It is the responsibility of the whole team to ensure that each team member completes their part of the work so that the overall software development project is successfully completed.

Accountability between developers and testers is usually very poor because when bugs are found in the software the, developers shift the blame to the business analyst who drew up the specification (Lagestee, 2013). Claiming that if the functionality was clearly defined in the specification there would be no bugs in the software. While testers are quick to blame the developers when the software has bugs, claiming that they developed code which is broken and malfunctioning (McHugh, Conboy, & Lang, 2012).

Both the developers and the testers need to realize that because they are part of one team, they are both accountable for the end product. Therefore, they need to work together to ensure that the quality of the software is not questionable. The onus is on the collective and not on an individual (McHugh, Conboy, & Lang, 2012).

2.4.6 Conflicting Personalities

Robles (2012) describes interpersonal skills as the life skills required for an individual to be able to effectively converse and relate with other people whether in the workplace, in relationships or in social settings. Organizations thrive on employing individuals with strong interpersonal skills so that they can get along with other employees, clients and customers as this has an impact on productivity and employee morale (Faheem, Fernando, Salah, & Piers, 2013). Furthermore, according to Faheem, Fernando, Salah, & Piers (2013) different individuals have different personalities and capabilities. Therefore assuming that a tester is fit to perform the work of the developer and vice versa would be unfair. A person's personality usually determines what they are capable of doing and this is referred to as their soft skill.

In a software development teams various individuals are involved in performing the tasks at the different stages of the software development life-cycle. In most cases the individuals choose to get involved in the tasks which best match their soft skills. "Individual differences in personality can explain and predict how judgments are made and how decisions are evaluated in software development projects" (Faheem, Fernando, Salah, & Piers, 2013: 173).

Faheem et al. (2013) further posit that developers are not very good in communicating whether with other team members or end-users. They typically prefer to work individually and tend to be more introverted. Developers tend to have very strong technical skills in that they are able to effectively and efficiently develop software according to specification but their soft skills tend to be very poor. Research shows that people who have strong soft skills tend to be more succesful than individuals who merely possess technical skill (O'Boyle, et al., 2011). Therefore developers need to improve on their soft skills in order to contribute to the overall success of a software development project.

Testers tend to have very strong soft skills in that they communicate well with their counterparts, they are willing to collaborate with developers in order to successfully complete a project and they are more extroverted. Their technical skills are limited but they are efficient and effective in the work that they do in spite of not being highly technical because they have problem solving and analytical skills. Both developers need to have a good balance of soft skills and technical skills because these two skill-sets complement each other (Faheem et al., 2013).

Differences in personality aggravate the misalignment between developers and testers. Developers usually struggle with human interaction, participation, communication and engaging other people in the work they do. Testers find it easier to work collaboratively, communicate regularly and engage in discussions while they work. These differences in interpersonal skills have a negative impact on the working relationship between developers and testers and thus affect the quality of the work they produce (Moeller & Zhang, 2012).

2.5 The Research Model

The research model that was used for this study is adopted from the factors discussed in the previous section which influence the misalignment between developers and testers. The model illustrates the relationship between the independent variables which are populated on the left of the diagram and the dependant variable populated on the right of the diagram. This model was chosen because it is the most applicable to the phenomena that is being investigated compared to other models that address lack of alignment and the model is context specific because it draws factors that affect individuals working in an agile SW development team.

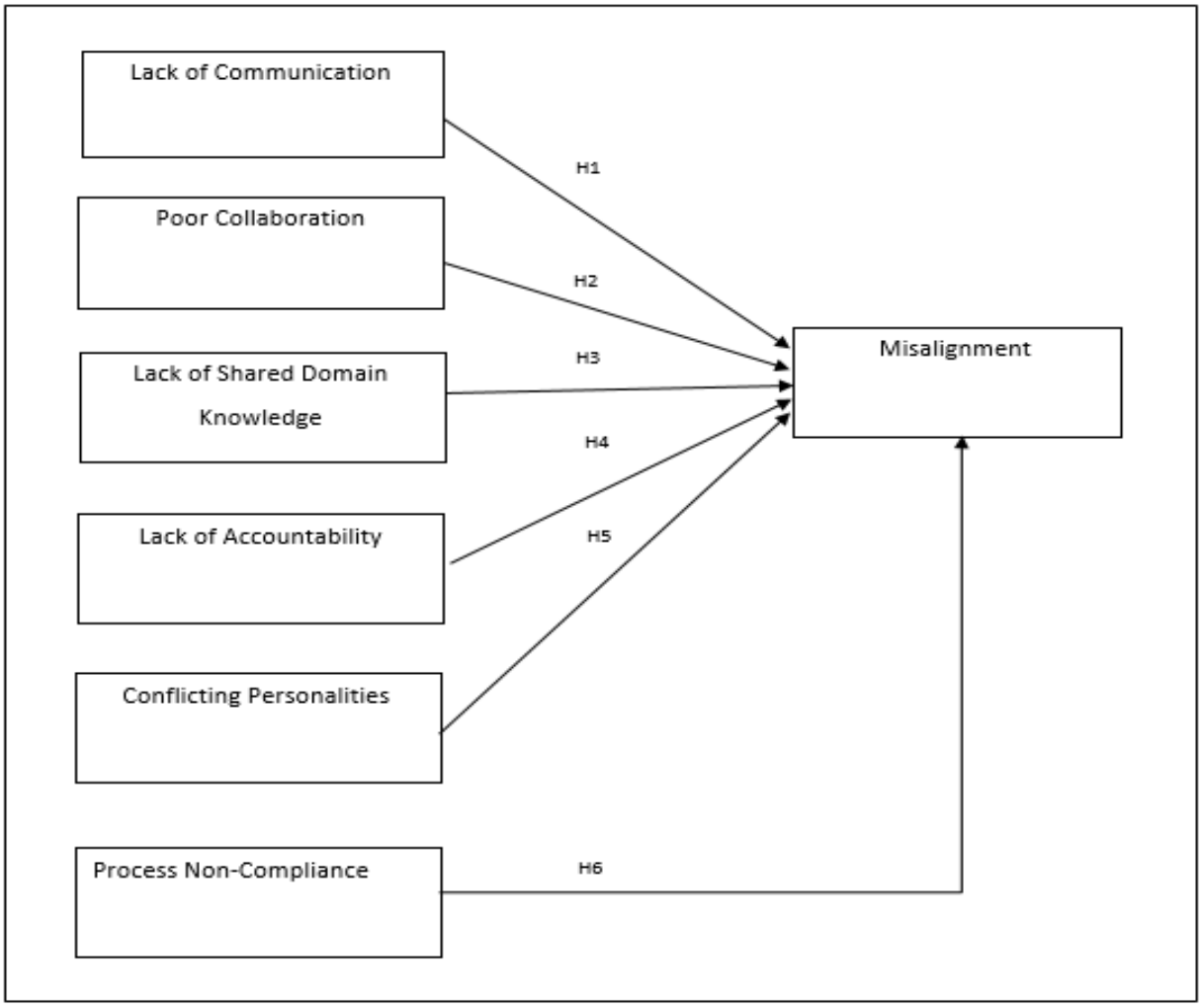

Figure 2 Conceptual Model

2.5.1 Hypotheses Development

The hypotheses depicted in Figure 2 are discussed below.

- Lack of Communication and Misalignment

According to Guimaraes, Staples, & McKeen (2004) effective communication plays a very important role in the productivity of software development projects and in the collaboration of the parties involved in the project and since developers and testers work very closely during the software development process. The developers need to transfer their knowledge of the software to testers while testers also need to communicate how they understand the functionality of the software to the developers. Thus this transfer of knowledge and information cannot be possible without effective communication between these two parties. If the communication is ineffective, this will result in a misalignment between the two parties. This notion results in the following hypothesis:

H1: Lack of Communication positively influences misalignment

- Poor Collaboration and Misalignment

Poor collaboration is defined as the lack of interaction with people resulting in the failure to accomplish a specific task (Brown, Lindgaard, & Biddle, 2011). Thus, poor collaboration between two or more parties' results in misalignment and the hypothesis below is supported:

H2: Poor Collaboration positively influences misalignment

- Lack of SDK and Misalignment

Maharaj & Brown (2015) posit that knowledge sharing are the set of activities that involve the transmission of information or help to others therefore SDK derives its definition from the former. Hence, knowledge sharing and shared knowledge are both relevant when referring to SDK. Thus, lack of SDK between developers and testers results in misalignment and the hypothesis below is supported:

H3: Lack of shared domain knowledge (SDK) positively influences misalignment

- Lack of Accountability and Misalignment

According to Wood & Winston (2007) accountability involves the acceptance of responsibility. Responsibility and accountability therefore go hand in hand. Maintaining a sense of accountability between developers and testers results in alignment between the two parties and the opposite is true when there is a lack of accountability thus the hypothesis below is supported:

H4: Lack of accountability positively influences misalignment

- Conflicting Personalities and Misalignment

Different individuals usually have different personalities and capabilities.Therefore assuming that a tester is fit to perform the work of the developer and vice versa would be unfair. A person's personality usually determines what they are capable of doing and this is referred to as their soft skill (Faheem, Fernando, Salah, & Piers, 2013). Furthermore, developers tend to be more introverted and poor in communication than their counterparts who are more extroverted and more vocal hence resulting in the misalignment between these two parties (Faheem, Fernando, Salah, & Piers, 2013). This results in the hypothesis below:

H5: Conflicting personalities positively influence misalignment

- Process Non-compliance and Misalignment

This type of alignment focuses on what organizational plans entail and the methods implemented to carry out these plans (Campbell, Kay, & Avison, 2005). When developers and testers do not know software development processes and adhere to these processes, this results in a misalignment between these two parties. The hypothesis below is thus supported.

H6: Process non-compliance positively influences misalignment

3. CHAPTER THREE: THE RESEARCH METHODOLOGY

3.1 Introduction

The following section aims to discuss the research philosophy and paradigm, research approach, research methodology, development of measures for research constructs, sample strategy, data collection method, data analysis, the research time frame and ethical considerations.

3.2 Research Philosophy and Paradigm

According to Myers (1997) all research, whether qualitative or quantitative is based on some philosophical assumptions on what makes up an acceptable research and the research methods that the research will assume. The underlying philosophy of this research is positivism. Positivistic researchers believe that facts can only be trusted once they have been verified through observation and measurement. In this philosophical stance, the researcher is limited to data collection and interpretation of the data and it is important that the researcher remains objective. The research findings for this stance are visible and measurable. In positivism studies, there is no room for researcher's human interests - the researcher needs to separate themselves from the study and remain as objective as possible (Crowther & Lancaster, 2008).

3.3 Research Approach

The research takes a deductive approach to theory as it commences by establishing a theoretical framework based on the available literature followed by data collection by method of surveys which will then be analyzed and measured in context of this framework (Saunders, Lewis, & Thornhill, 2011). The following stages are followed in the deductive approach:

1. Deduce hypotheses from a theory/framework/model
2. Develop hypotheses and suggest a relationship/s between the independent and dependent variables
3. Test the hypotheses against the data that emerges from data collection
4. Analyze the data and the outcome of the test to accept or reject the proposed theory/framework/model
5. Adjust the theory/framework/model based on the outcome of the hypotheses tests

The research aims to investigate the underlying factors that result in the misalignment between developers and testers and it seeks to go beyond exploring and describing the problem, therefore the research is explanatory in nature. The problem being investigated is known and has been researched before but from a different perspective of functional alignment (Marczyk, DeMatteo, & Festinger, 2005), therefore cannot be described as exploratory. There is a need to investigate what causes social misalignment hence the need to conduct this explanatory research (Joseph & Kavita, 2008). Explanatory research is often quantitative in nature and usually tests prior hypotheses by measuring relationships between variables. The data is subsequently analyzed using statistical methods.

3.4 Research Methodology

3.4.1 Development of Measures for Research Constructs

The following section aims to discuss how the constructs that arise in the hypotheses section were measured. In this section these factors will be referred to as constructs because they form the basis for the framework that will be developed for the study. In this section, the instruments for measuring each construct are developed and discussed. Each construct is defined and explained, followed by a table which outlines the variables that will be used to measure the constructs. These variables form the basis of the research questionnaire that was used for data collection (See Appendix C).

A five-point Likert scale ranging from 1 for ‘strongly agree’ to 5 for ‘strongly disagree’ was used in the questionnaire for all the items. According to Nemoto & Beglar (2014) the use of Likert scales allows for the quick distribution of questionnaires to many respondents, the validity of the analysis made from the data can be tested through a variety of Means and the quantitative data can be matched and used in conjunction with other qualitative data-collection techniques as would be the case in a mixed methods research.

3.4.1.1 Dependent Variable

Misalignment

Throughout the paper, a standard definition of misalignment is applied, this definition is derived from Hanson, Melnyk, & Calantone (2011) who define alignment as a condition that can be created, one that directly influences competitive advantage and encourages teamwork and harmony within the organization. This type of alignment encourages understanding of the project strategy, acceptance of the project strategy, having a link between combined efforts in the team and setting standards/targets that need to be met by the team. Thus, this definition is applied to the current study as it addresses the social aspects of alignment in the context of developers and testers.

The measurement for misalignment was derived from Hanson, Melnyk, & Calantone (2011: 1103-1109). While Hanson et al. (2011) consider alignment with respect to strategic and performance management, this study applies the measure to misalignment in the agile software development context. Hanson et al. (2011) include understanding, acceptance, linkage, consistency, standards and incentives as dimensions of alignment. Measurement items for our misalignment construct included four of these dimensions, i.e. understanding, acceptance, linkage and standards.

The following items, adapted from Hanson, Melnyk, & Calantone (2011) are proposed to measure misalignment between developers and testers in agile organizations:

1. Understanding – it is important for developers and testers to understand the project strategy
2. Acceptance – once the project strategy is understood then the developers and testers need to accept and adhere to it.
3. Linkage –There need to be a clear link between their combined efforts and the overall outcome of projects.
4. Standards – reasonable targets need to be set for software development teams and the success of these targets should be a team effort and not an individual effort.

These measures are shown in Table 1 below.

Table 1 Dependent variable - Misalignment

Misalignment		
Item	**Variable**	**Source**
MISAL1	Developers and Testers do not have a shared understanding of agile software development projects	(Hanson, Melnyk, & Calantone, 2011)
MISAL2	Developers and Testers do not adhere similarly to agile software development projects	(Hanson, Melnyk, & Calantone, 2011)
MISAL3	Developers and Testers do not have a clear link between their combined efforts and outcome of a project	(Hanson, Melnyk, & Calantone, 2011)
MISAL4	Developers and Testers do not collectively set standards and targets which they need to meet	(Hanson, Melnyk, & Calantone, 2011)

3.4.1.2 Independent Variables

Lack of Communication

Guimaraes, Staples, & McKeen (2004) posit that the manner in which developers and testers communicate psychologically affects the software development project. When communication is ineffective between the two parties, there is a high possibility that there will be misunderstanding on how the software is perceived and how it actually works by both the parties.

The measurement for lack of communication was derived from Guimaraes, Staples, & McKeen (2004). The model in this paper is for measuring the variables that influence system quality between the end-user and the developer. Communication is illustrated as a variable that affects system quality between two parties in a software development project. The arguments highlighted in the paper were used to extract a suitable variable that can be used to measure lack of communication between two parties in a software development project i.e. the software tester and developer. The variables below in Table 2 demonstrate how to measure the lack of communication between developers and testers.

Table 2 Independent variable - Lack of communication

Lack of Communication		
Item	**Variable**	**Source**
COMM1	There is a lack of a common language between Developers and Testers	(Guimaraes, Staples, & McKeen, 2004)
COMM2	Developers and Testers do not listen to each other	(Guimaraes, Staples, & McKeen, 2004)
COMM3	Developers and Testers do not express their ideas clearly to each other	(Guimaraes, Staples, & McKeen, 2004)

Poor Collaboration

Mellin, et al. (2010) identify four elements which are prevalent in situations where there is poor collaboration. These include lack of interdependence, where there is lack of interaction between developers and testers for the completion of tasks and activities. Lack of flexibility is the second element where the role of the developer and tester are very strict and there is no flexibility in the tasks that they can and cannot perform. The next element is the lack of collective ownership of goals, this involves both developers and testers taking a shared responsibility of the team goals.

The measurement for poor collaboration was derived from Mellin, et al. (2010). The paper is written to measure interprofessional team collaboration in expanded school mental health. In this paper Mellin, et al. (2010) describes five factors that need to be considered for effective collaboration:

- Interdependence – this factor infers that collaborators rely on interactions with other professionals to accomplish their goals and tasks.
- Newly created professional activities – this factor refers to collaborative acts, programs and structures that allow for goals to be accomplished collectively which could not be achieved independently.
- Flexibility – refers to the deliberate blurring of professional roles.
- Collective ownership of goals – this refers to sharing of responsibility and joint effort in the process of achieving goals.
- Reflection on Process – this refers to collaborators being intentional about committing to the process of working together and the outcomes of the work.

Therefore, the measurements for poor collaboration were derived from the arguments raised in this paper. This leads to the following item measures in Table 3.

Table 3 Independent variable - Poor Collaboration

Poor Collaboration		
Item	**Variable**	**Source**
COLLAB1	Interaction between Developers and Testers is minimal	(Mellin, et al., 2010)
COLLAB2	Mutual respect between Developers and Testers is lacking	(Mellin, et al., 2010)
COLLAB3	Co-operation between Developers and Testers is poor	(Mellin, et al., 2010)
COLLAB4	Collective ownership of goals between Developers and Testers is lacking	(Mellin, et al., 2010)

Lack of Shared Domain Knowledge (SDK)

Three measurements for lack of shared knowledge were derived from Espinosa, Slaughter, Kraut, & Herbsleb (2007). The paper addresses the importance of coordination between geographically dispersed members in software development teams through knowledge sharing. The paper infers that developers usually produce effective software when they operate individually but the software may fail when integrated with the code of other developers because it was not tested correctly. The first measurement (SDK1) was derived from this argument. The paper further posits that developers are only concerned about what is tested and by whom when they need feedback on the outcome of the test but overall they are ignorant of such information. Thus, the second measurement was derived (SDK2). Moreover, the paper highlights lack of presence awareness as a problem that affects effective coordination in software development teams. This statement is the basis for the third measurement (SDK3). Thus, the above arguments are used to derive the first three measurements.

The next three measurements are derived from Arnicane (2007). In this paper, Arnicane highlights that testers have limited knowledge on how to develop code and the testing done by developers and limited technical skills. The arguments discussed in this paper are used as the basis to derive the last three measurements (SDK4, SDK5 and SDK6). The measures in Table 4 below illustrate how the lack of SDK can be measured between developers and testers in agile organizations.

Table 4 Independent variable - Lack of shared domain knowledge

Lack of Shared Domain Knowledge		
Item	**Variable**	**Source**
SDK1	Developers lack knowledge on how to thoroughly test software	(Espinosa, Slaughter, Kraut, & Herbsleb, 2007)
SDK2	Developers lack knowledge on the value added by Testers in an agile software development team	(Espinosa, Slaughter, Kraut, & Herbsleb, 2007)
SDK3	Developers do not have strong soft skills	(Espinosa, Slaughter, Kraut, & Herbsleb, 2007)
SDK4	Testers lack knowledge on how to resolve code issues	(Arnicane, 2007)
SDK5	Testers lack knowledge on the type of testing done by developers	(Arnicane, 2007)
SDK6	Testers lack technical software development skills	(Arnicane, 2007)

Lack of Accountability

According to Wood & Winston (2007) there are four elements which can be used to measure the lack of accountability. The first element is lack of responsibility, this means that developers and testers struggle to take responsibility when issues arise in the projects they are responsible for. Secondly, lack of openness refers to the lack of transparency of the goals, visions and tasks that are performed by the developers and testers. Lastly, lack of answerability means that developers and testers do not admit to mistakes they have made in order to subsequently formulate strategies for alleviating previous mistakes from occuring again.

The measurement for lack of accountability was derived from Wood & Winston (2007). The measurements for accountability that are discussed in the paper are relevant to individuals in leadership roles but were made applicable to the agile software development context. The measures were then reworded using the arguments raised in the paper to derive the measurements for lack of accountability. The measures discussed by Wood & Winston (2007) include willing acceptance of responsibilities, public dislosure of words and actions and answerability for beliefs, decisions, commitments and actions. These measurements are illustrated in Table 5 below.

Table 5 Independent variable - Lack of accountability

Lack of Accountability		
Item	**Variable**	**Source**
ACC1	Developers and Testers do not take joint responsibility for their actions in a team	(Wood & Winston, 2007)
ACC2	There is lack of transparency between the work of Developers and Testers	(Wood & Winston, 2007)
ACC3	There is a lack of openness between Developers and Testers	(Wood & Winston, 2007)
ACC4	Developers and Testers fail to jointly account for the outcomes of their actions and decisions	(Wood & Winston, 2007)

Conflicting Personalities

The measurement for conflicting personalities (INTPERS1, INTPERS3) was derived from Alge, Gresham, Heneman, Fox, & McMasters (2002). In this paper the authors investigate whether interpersonal skills diretly affect customer service. The variables that discuss interpersonal skills, extroversion and general disposition were derived and modified so that they can be applicable to the relationship between developers and testers. The reasoning behind using this paper was because of the emphasis made in the discussion on the relationship between interpersonal skills and performance.

Furthermore, a paper by Baron & Tang (2009) was used to derive one of the measurement for conflicting personalities. The authors in this paper higlight the importance of social skills between parties that interact with each other in a professional environment. The arguments raised in this paper together with the discussion relating to collaboration between designers and developers in a paper by Brown, Lindgaard, & Biddle (2011) were used as premise to derive (INTPERS2). The measurements for conflicting personalities are illustrated in Table 6 below.

Table 6 Independent variable - Conflicting personalities

Conflicting Personalities		
Item	**Variable**	**Source**
INTPERS1	Testers tend to be more extroverted than developers	(Alge, Gresham, Heneman, Fox, & McMasters, 2002)
INTPERS2	Testers tend to be more diplomatic than developers	(Baron & Tang, 2008)
INTPERS3	Developers and Testers usually do not get along	(Alge, Gresham, Heneman, Fox, & McMasters, 2002)

Process Non-Compliance

For developers and testers to have a good working relationship and be effective as a team, they need to comply with the process/s that underpin an agile environment. Kohlbacher & Gruenwald (2011) identify a number of elements which are responsible for the process non-compliance in agile organizations.

The measurements for process non-compliance were derived from Kohlbacher & Gruenwald (2011). The paper investigates the dimensions that shape process orientation in organizations and their effectivess in improving an organizations performance. The paper highlights the following dimensions:

- process design and documentation;
- management commitment;
- process owner;
- process perfromance measurement;
- corporate culture in line with the process approach;
- people and expertise; and
- coordination and integration of process projects.

The measurements for this construct were subsequently derived from the arguments that evolved from the discussion of the above process orientation dimensions and made applicable to the agile software development context. The measurements for process non-compliance are illustrated in Table 7 below.

Table 7 Independent variable - Process non-compliance

Process Non-compliance		
Item	**Variable**	**Source**
PROCESS1	Agile software development processes are not clearly designed	(Kohlbacher & Gruenwald, 2011)
PROCESS2	Agile software development processes are not clearly documented	(Kohlbacher & Gruenwald, 2011)
PROCESS3	Process performance measurements are poorly conducted in agile software development teams	(Kohlbacher & Gruenwald, 2011)
PROCESS3	There is a lack of process improvement skills among Developers and Testers	(Kohlbacher & Gruenwald, 2011)

3.5 Research Sample Strategy

3.5.1 Sampling Technique

The data sampling method is purposive sampling. In purposive sampling a specific group of people is chosen within a population. The aim of purposive sampling is to choose a sample which possesses certain characteristics that are relevant to the research (Tongco, 2007). The selected population must work in an organization where the agile methodology is being applied or some of the principles of this methodology are being implemented to manage software development projects.

3.5.2 Population and Frame

In this case, instead of choosing all individuals that make up a software development team, only developers, testers and IT managers were selected. The advantage of also collecting data from IT managers will ensure that there is no bias data and that data is also obtained from an external party other than the developer or tester. IT managers work very closely with both parties and can provide objective data to analyze this phenomenon. These respondents should have been exposed to one or more software development projects during their time in that organization.

There are no gender limitations in the data collection and there is no specific race targeted for the study. The respondents do not have to be any age as long as they meet the above criteria related to the role, term of service and exposure to project work in a team.

A sample size large enough to lead to statistically significant results is necessary. Given there are seven constructs in the research model a minimum sample size of 70 was required (Ritchie, Lewis, & Elam, 2003).

The same number of developers, and testers is ideal so that there is an equal split between these parties, with an additional set of IT managers to achieve triangulation of perspectives (Marczyk, DeMatteo, & Festinger, 2005). This will guarantee that the views of the developers, testers are equally presented, as well as the views of IT managers. Thus, the aim was to obtain 70 questionnaire responses. 30 questionnaires to be completed by developers, another 30 to be completed by testers and the remaining 10 to be completed by IT managers. The reason for choosing a small sample of IT managers is in accordance with Marczyk, DeMatteo, & Festinger (2005) who state that purposive sampling is at the researchers' discretion. The researcher purposely decides who to include in the study based on the respondents' ability to provide the required data. As mentioned previously, the study aims to investigate misalignment between developers and testers hence these form a majority of the sample. IT managers are included for triangulation.

3.5.3 Data Collection

A survey questionnaire was used to gather the quantitative data. The advantage of a survey questionnaire is that it allows the researcher to investigate multiple variables at once and this is not possible through other quantitative research methods such as laboratory or field experiments (Brown & Jayakody, 2008). Another reason for selecting this method of data collection is because the framework has been successfully established through literature and this will be the basis for the quantitative analysis. One of the disadvantages of using a survey is that the researcher cannot establish the perceptions and motives that lead to the respondents affecting the phenomena that is being investigated (Tan & Teo, 2000).

The data was collected using a survey that was populated on third-party online software called Qualtrics. The survey was launched onto the Qualtrics website after extensive communication with the Qualtrics representatives to ensure that the format, structure and content of the survey was satisfactory. The advantage of using this tool for distributing the survey was that it can only be taken by individuals who were invited to participate therefore preventing individuals who do not form part of the sample to participate (Pennington & Kelton, 2016). Furthermore, the tool also allows for participants in various countries to participate such as United States of America (USA) in the case of this study.

A two-phase approach was used to collect data, to increase generalizability of findings. In Phase 1, emails were sent out to developers, testers and IT managers in South Africa (SA) with a link to the questionnaire on Qualtrics. The link to the questionnaire was also shared on social media and social networking websites to the relevant parties who form part of the target population. A pilot questionnaire was sent out to a few testers, developers and IT managers prior to launching the final survey on Qualtrics as a form of pilot test. This was done to ensure that the questionnaire was user friendly and made sense to the users so that it yields effective results. All the parties involved in the pilot test found the questions straight forward and understandable, therefore the questionnaire was deemed ready for distribution and for data collection to commence.

Due to the poor response from respondents in SA, the researcher sent a request to the Qualtrics team to assist with a panel of respondents to participate in the data collection. The Qualtrics team recruited a panel of respondents who met the requirements as set out in the sampling strategy. Most the panel of respondents was from the USA hence the need to categorize the responses into SA and USA. This formed phase 2 of the data collection process. Such a process adds validity to the study, as exemplified by Pennington & Kelton (2016).

A confidentiality clause was included in the survey stipulating that responses will be handled with confidentiality and respondents can provide their contact details if they wish to receive findings of the study. Cover letters were sent out to the participants that will be selected for data collection. No organization permission was required because the data was collected from IT professionals as respondents and these professionals were directly approached by the researcher. The sample cover letter is included in Appendix B of this report.

3.5.4 Data Analysis

Data analysis was performed through Statistica software. This type of software caters for data analysis, data management, data mining and data visualization procedures amongst others (Weiss, 2008). The data was imported from a spreadsheet after the necessary formatting was completed. The following data analysis tests were performed using Statistica software:

- Confirmatory Factor Analysis – prior to hypothesis testing, the research instrument was tested for validity (Tan & Teo, 2000). All items with factor loadings < 0.6 were eliminated as well as items that loaded on more than one factor. Certain constructs were merged to form one construct where they loaded together if there was justification for this occurrence. This process was performed until validated constructs were developed (Maharaj & Brown, 2015).
- Cronbach's Alpha - Once all the constructs that develop have been validated, the Cronbach's alpha was calculated to test the reliability of the constructs that emerged (Tan & Teo, 2000). For a construct to be deemed as reliable, it must have a Cronbach's alpha that is greater than 0.7 (Maharaj & Brown, 2015).
- T-tests – t-tests were conducted to evaluate if there was any significant difference between the Means of two groups (Trochim, Donnelly, & Arora, 2015). In this case, t-tests were computed to evaluate the Means between responses from SA and USA as well as responses from developers and testers.
- Spearman's Rank Correlation Coefficient – this non-parametric rank statistic was used to measure the strength of association between two variables (Hauke & Tomasz, 2011). This test was conducted to establish if there is correlation between the independent variable and the dependent variables – i.e. to test hypotheses.
- ANOVA test - The ANOVA test is useful in determining differences in the Means of three or more independent/unrelated groups (Vijayvargiya, 2009) and was used to evaluate the difference in Mean scores between developers, testers and IT managers for all the relevant constructs and demographic information.

3.5.5 Triangulation and Validation

Usability – It is important that the researcher chooses an instrument that will be easy to manage and understand for themselves and the participants otherwise they may cause confusion and not get the correct data from the instrument or the participants. It is important to consider the following points to ensure that the chosen instrument is user friendly: 1) the length of time it will take to oversee the use of the instrument. 2) Clarity of directions to use the instrument. 3) How simple or complex it is to score the instrument. 4) Any problems that have been conveyed in the use of the instrument. It is always advisable to use an instrument that has been tried and tested to avoid any unnecessary problems and time wasting (Biddix, 2014).

Validity – When choosing a research instrument, it is imperative to choose an instrument that does what is expected to do. If it is meant to measure, it should measure and not estimate. Finding an instrument that will be 100% accurate is almost impossible but the validity of the chosen instrument should be almost perfect (Biddix, 2014). Confirmatory factor analysis was used to test the validity of the research instrument. The factors were refined through construct validity and face validity until all the constructs demonstrated acceptable construct and discriminant validity (Maharaj & Brown, 2015).

Reliability – The chosen instrument should be dependable; it should accurately do what it is intended to do. Reliability cannot be calculated but can be estimated through observations and tests. For example, to test reliability, one can evaluate the degree to which observers give consistent responses or one can conduct the exact same tests in two different settings using the same data in both settings (Biddix, 2014). The Cronbach's alpha was calculated for each construct to test the reliability of the refined instrument (Maharaj & Brown, 2015).

3.5.6 Research Timeframe

A cross-sectional timeframe was applied in the study because this type of method allows for the comparison of different population groups at the same time (Levin, 2015). This allows for developers, testers and managers to be compared at one period. Furthermore, this method allowed the researcher to compare several variables at once. Therefore, allowing the researcher to investigate how the independent variables affect the dependent variable at the same time instead of investigating the constructs one by one over time (Levin, 2015).

3.5.7 Ethical Considerations

Involvement of the participants in the research was voluntary and the researcher handled the data collected from the respondents with the utmost confidentiality. The names of the individuals that were investigated in the study were kept anonymous. The researcher obtained approval from the University of Cape Town to carry out the research and the researcher adhered to the University's ethical requirements when conducting the research. Please see Appendix A for the ethics approval letter.

4. CHAPTER FOUR: RESEARCH ANALYSIS, DISCUSSION AND FINDINGS

4.1 Introduction

The following section illustrates the demographic profile of the respondents that participated in the data collection. This is followed by construct and discriminant validity, instrument reliability analysis, t-tests, ANOVA test and Spearman's rank extracted from Statistica software to illustrate the findings of the data analysis. The refined conceptual model is then illustrated and discussed in the final section of the chapter.

4.2 Data Analysis and Findings

4.2.1 Demographic Profile

The demographic profile of the respondents that participated in the study is illustrated in Table 8 below. A total of 47 responses were received from the two phases of data collection. 23 of the respondents were from SA and the remaining 24 respondents were from the USA.

The demographic profile illustrates that 49% of the sample was composed of testers, 32% were developers, 17% were IT managers and 2% were categorized as other. 51% of the respondents have more than 5 years of experience in their role and approximately 49% of the respondents have more than 5 years of experience in agile software development. Furthermore, 49% have been involved in more than 5 software development projects.

Table 8 Demographic profile

Item	Value	%
Role		
Developer	15	31,9
Tester	23	48,9
Manager	8	17,0
Other	1	2,1
Number of years in role		
<1	3	6,4
1-3	6	12,8
3-5	14	29,8
5-7	12	25,5
>7	12	25,5
Agile software development experience		
<1	4	8,5
1-3	8	17,0
3-5	12	25,5
5-7	9	19,1
>7	14	29,8
Number of software development projects done		
<1	4	8,5
1-3	8	17,0
3-5	12	25,5
5-7	9	19,1
>7	14	29,8

4.2.2 Construct Validity

The data was assessed for construct validity using confirmatory factor analysis prior to hypothesis testing (Tan & Teo, 2000). To conduct this test, the data was imported into Statistica (StatSoft, 2016).

All the item measures from the 7 constructs were selected for factor analysis. The total number of factors expected was specified as 7 and the minimum eigenvalue for extraction was set at 1. The factor rotation selected was varimax normalized and the factor loadings were selected to be greater than 0.6 (Brown & Jayakody, 2008). All factor loadings less than 0.6 were removed because these were deemed to have poorly loaded or are unacceptable. Items which loaded on a factor that was not related to the item were also removed. Face validity and construct validity were used to analyze the factor loadings and to gradually refine the research instrument. Table 9 below illustrates the results of the factor loadings for remaining items after a series of deleting ill-fitting items.

Table 9 Final factor loadings

ITEMS	ACC/PROCESS	TKD	PERS	COLLAB	DKT	MISALIGN
MISAL3	0,34	0,06	0,35	0,26	0,09	**0,70**
MISAL4	0,34	0,12	-0,03	0,22	0,08	**0,79**
COMM3	0,46	0,11	0,15	**0,75**	0,09	-0,05
COLLAB1	0,27	0,23	0,02	**0,85**	0,03	0,23
COLLAB3	0,36	0,19	0,11	**0,69**	0,07	0,31
COLLAB4	**0,68**	0,18	-0,04	0,25	0,24	0,27
SDK1	0,51	0,01	-0,01	0,16	**0,65**	-0,09
SDK3	0,17	0,18	0,20	0,05	**0,79**	0,09
SDK4	0,23	**0,73**	-0,16	0,18	0,40	0,08
SDK5	0,11	**0,78**	0,17	0,09	-0,07	0,39
SDK6	0,16	**0,87**	0,08	0,14	0,03	-0,14
ACC1	**0,76**	0,32	0,17	0,11	0,10	0,13
ACC2	**0,77**	0,11	0,12	0,29	0,15	0,18
ACC3	**0,76**	0,27	0,00	0,39	0,08	0,15
ACC4	**0,87**	0,13	0,19	0,16	0,02	-0,01
INTPERS1	0,13	0,10	**0,92**	0,14	0,02	-0,02
INTPERS2	0,10	-0,03	**0,70**	0,00	0,49	0,27
INTPERS3	0,09	-0,01	0,05	**0,64**	0,57	0,23
PROCESS1	**0,70**	0,00	-0,06	0,30	0,28	0,37
PROCESS2	**0,73**	0,00	-0,06	0,27	0,16	0,39
PROCESS3	**0,66**	0,02	0,26	0,02	0,38	0,25
Explained variance	5,48	2,30	1,75	2,89	2,14	2,09
Proportional total	**0,26**	**0,11**	**0,08**	**0,14**	**0,10**	**0,10**

- Lack of Accountability/Process Non-Compliance (ACC/PROCESS)

Table 9 illustrates that Factor 1 consisted of items COLLAB4, ACC1 to ACC4 and PROCESS1 to PROCESS3. All items were checked to see if on face value they belong together. COLLAB 4 which denotes poor collaboration resonated with the Lack of Accountability (ACC1, ACC2, ACC3, and ACC4) therefore all these items were retained. The Lack of Accountability items and Process Non-Compliance items (PROCESS1, PROCESS2, and PROCESS3) were on face value different but since they loaded on one construct, to remain true to the data, they were all retained. This resulted in all these items being grouped because they were all valid with factor loadings >0.6. Thus, Factor 1 was renamed to ACC/PROCESS.

- Lack of Testers Knowledge about Development (TKD)

SDK4, SDK5 and SDK6 loaded on Factor 2. On face value, it is evident that these constructs belong together as they referred to the tester's knowledge of development and loaded on the same factor and were subsequently retained. This resulted in factor 2 being renamed to TKD.

- Lack of Interpersonal Skills (INTERPERS)

INTERPERS1 and INTERPERS2 loaded on Factor 3 resulting in factor 3 being renamed to PERS.

- Poor Collaboration (COLLAB)

Factor 4 was made up of COMM3, COLLAB1, COLLAB3 and INTERPERS3. The items were closely related and all referred to the working relationship between developers and testers hence the respondents viewed these variables as one. Factor 4 is then renamed to COLLAB.

- Lack of Developer Knowledge of Testing (DKT)

Factor 5 consisted of SDK 1 and SDK3 causing this factor to be renamed to DKT as the items all refer to the lack of developer's knowledge of testing.

- Misalignment (MISALIGN)

Lastly, factor 6 loaded with items MISAL3 and MISAL4 resulting in this factor to be named MISALIGN as it is related to questions that refer to misalignment between developers and testers.

This process of refining, renaming and removing certain constructs resulted in construct and discriminant validity (Maharaj & Brown, 2015).

4.2.3 Reliability Analysis

The reliability test for each variable was conducted by computing the Cronbach's alpha value for each validated construct in Statistica software. The Cronbach's alpha is a popular method used to determine the reliability of the refined instruments when working with Likert scales (Tan & Teo, 2000). The Cronbach's alpha must be greater than 0.7 for all the factors to be observed as reliable (Maharaj & Brown, 2015). Table 10 below is an illustration of the reliability results generated by Statistica. The Cronbach's alpha for all the validated constructs is greater than 0.7 and thus it can be concluded that the refined constructs are all reliable because they yield a Cronbach's alpha that is greater than 0.7.

Table 10 Reliability Analysis

Factor	Cronbach's Alpha
Lack of Accountability and Process Non-compliance (ACC/PROCESS)	0,72
Tester Knowledge of Development (TKD)	0,79
Interpersonal Conflict (PERS)	0,81
Poor Collaboration and Lack of Communication (COLLAB)	0,76
Developer Knowledge of Testing (DKT)	0,72
Misalignment (MISALIGN)	0,75

4.2.4 T-test – USA vs South Africa

Table 11 below compares the Mean scores between USA and SA for term (years) in role, management experience in years (Mngt Exp), agile software development experience in years (Agile SW Dev Exp), number of agile software development projects experienced (Agile SW Dev Proj), misalignment (MisAlign), poor collaboration (Collab), lack of developer knowledge of testing (DKT), lack of tester knowledge of development (TKD), conflicting interpersonal skills (Pers) and lack of accountability and process non-compliance (Acc/Process).

The variables were selected as shown below with the grouping variable being the Country. The independent-samples t-test is effective in evaluating the Means for two unrelated or independent groups (VanVoorhis & Morgan, 2007). The table 11 below illustrates the output of the independent-samples t-test.

Table 11 T-test for USA vs SA

Factor	Mean USA	Mean SA	t-value	Df	p	Valid N	Valid N
Term in role	3,74	3,29	1,29	45	0,20	23	24
Mngt Exp	2,13	2,33	-1,17	45	0,25	23	24
Agile SW Dev Exp	2,91	3,25	-1,01	45	0,32	23	24
Agile SW Dev Proj	3,57	3,33	0,60	45	0,55	23	24
MisAlign	2,98	2,54	1,23	44	0,22	22	24
Collab	3,38	2,88	1,46	44	0,15	22	24
DKT	2,91	2,94	-0,08	44	0,93	22	24
TKD	2,62	2,85	-0,73	44	0,47	22	24
Pers	2,68	2,77	-0,29	44	0,77	22	24
Acc/Process	2,78	2,97	-0,63	44	0,53	22	24

Most Mean scores for USA are different from SA, but not significantly different at p = 0.05. Evidence of insignificance is seen in the p-value of the variables between the two groups being greater than 0.05. For the term in role, the Mean score for USA was higher than SA meaning that USA respondents had more experience in years in their roles than SA respondents. Furthermore, for agile software development projects experienced, the Mean score was higher for USA than for SA meaning that USA respondents had more experience in agile software development projects than SA respondents.

There is no statistically significant difference between the Mean scores of the dependent variables selected for the two groups, meaning that the respondents from USA and SA share similar views on the above factors therefore the geographic location of the respondents has no apparent influence on the two sets of data, thus the data can be combined.

4.2.5 T-test - Tester vs Developers

Independent-samples t-test were computed to compare the Means between developers (DEV) and testers (TEST) for term (years) in role, management experience in years (Mngt Exp), agile software development experience in years (Agile SW Dev Exp), number of agile software development projects experienced (Agile SW Dev Proj), misalignment (MisAlign), poor collaboration (Collab), lack of developer knowledge of testing (DKT), lack of tester knowledge of development (TKD), conflicting interpersonal skills (Pers) and lack of accountability and process non-compliance (Acc/Process).

The dependent variables were selected as shown in table 12 below with the grouping variable being the Role. Group 1 represents developers and group 2 represents testers.

Table 12 T-test for Testers vs Developers

Factor	DEV Mean	TEST Mean	t-value	df	p	Valid N	Valid N
Term in role	3,42	3,67	-0,63	37	0,53	24	15
Mngt Exp	2,08	2,27	-0,95	37	0,35	24	15
Agile SW Dev Exp	2,79	3,47	-1,89	37	0,07	24	15
Agile SW Dev Proj	3,25	3,53	-0,67	37	0,50	24	15
MisAlign	3,04	2,63	1,00	37	0,32	24	15
Collab	3,56	2,98	1,60	37	0,12	24	15
DKT	3,10	2,70	1,09	37	0,28	24	15
TKD	**3,14**	**2,40**	**2,32**	**37**	**0,03**	**24**	**15**
Pers	2,71	3,03	-0,95	37	0,35	24	15
Acc/Process	3,10	2,67	1,26	37	0,22	24	15

Where p > 0.05 it indicates that there is no statistically significant difference between the Mean scores of the dependent variables selected for the two groups. The Mean scores for Developers (DEV) are different from Testers (TEST), but not significantly different, meaning that one would assume that developers and testers are one group instead of two groups because there is no significant difference between the Means of the two groups. The exception was for TKD where there was a statistically significant difference (p <0.05) between the Mean scores of the developers and testers about the lack of testers knowledge of development (TKD) therefore developers (DEV) and testers (TEST) somewhat disagree on the extent to which there is lack of testers knowledge of development (TKD).

For the scores on misalignment (MisAlign), tester lack of knowledge of development (TKD), poor collaboration (Collab) and lack of developer knowledge of testing (DKT), testers scored lower than developers meaning that testers agree more with these factors than developers.

4.2.6 ANOVA Test

The one-way analysis of variance test was used to test if there are any statistically significant differences between the Means of the developers, testers and IT managers. The ANOVA test is useful in determining differences in the Means of three or more independent/unrelated groups and is thus the most effective method in this case because there are three unrelated groups (Vijayvargiya, 2009). Table 13 below illustrates the Means scores for all the independent and dependent variables as well as the Mean scores for the demographic profile to evaluate any statistically significant differences in the Means scores of the above-mentioned items between the developers, testers and managers.

Table 13 ANOVA Test

Items and Mean values	**MISALIGN**	**COLLAB**	**DKT**	**TKD**	**PERS**	**ACC/PROCESS**	**Term in role**	**Mngt Exp**	**Agile SW Dev Exp**	**Agile SW Dev Proj**	**Average Mean**
p-value	0,41	**0,00**	0,13	**0,02**	0,32	0,20	0,74	0,08	0,17	0,06	
Tester	2,63	2,97	2,70	2,40	3,03	2,66	3,66	2,26	3,46	3,53	2,93
Developer	2,95	3,56	3,19	3,17	2,65	3,13	3,43	2,08	2,82	3,17	3,02
Manager	2,28	2,00	2,28	2,19	2,35	2,46	3,75	2,62	3,37	4,37	2,77

There were no statistically significant differences between the Means of the three groups for the items as determined by one-way ANOVA for the following:

- Misalignment (MISALIGN)
- Lack of developer knowledge of testing (DKT)
- Lack of interpersonal skills (PERS)
- Lack of accountability and process non-compliance (ACC/PROCESS)
- Term in role (Term in role)
- Management experience (Mngt Exp)
- Agile software development experience (Agile SW Dev Exp)
- Number of agile software development projects experienced (Agile SW Dev Proj)

All the above items had a p-value > 0.05 as shown in the table above hence it is concluded that there were no statistically significant differences between the Means of developers, testers and managers for the items above. On the contrary, there were statistically significant differences between the Means of the three groups for poor collaboration (COLLAB) and lack of tester knowledge of development (TKD). These items had a p-value < 0.05 as shown in table 13. Developers had the highest Mean for poor collaboration (COLLAB) compared to testers and managers. This means developers disagree that there is poor collaboration between developers and testers while testers' and managers' responses indicate that they agree that collaboration is poor.

Furthermore, developers had the highest Mean for lack of tester knowledge of development (TKD). Meaning that the developers somewhat disagree that there is lack of tester knowledge of development (TKD) while testers and managers agree that there is lack of tester knowledge of development (TKD).

The average of the Means between the three groups illustrate that managers were more optimistic in their responses compared to testers who are the most pessimistic despite having the shortest term in the role, the least management experience, the least agile software development experience and the least agile software development project experience. Developers were also more optimistic in their responses compared to testers.

4.2.7 Spearman's Rank

To evaluate the hypotheses that were formulated in the study, a two-tailed non-parametric statistic, the Spearman's rank correlation coefficient association was used rather than multiple linear regression analysis as per Tan & Teo (2000), etc. In regression analysis, one needs 10 responses per independent variable but in this study, less than 10 responses were received per independent variable hence the use of Spearman's rank correlation coefficient (Hart & Gabi, 2004). Spearman's rank is used to test if there is a correlation between two sets of variables (Hauke & Tomasz, 2011).

A Spearman's rank of 0 indicates that there is no correlation between the two sets of rankings. While a Spearman's coefficient of 1 indicates that the two rankings are correlating perfectly. According to the illustration below, marked correlations are significant at $p < 0.05$.

The results of the Spearman rank are illustrated in table 14 below.

Table 14 Spearman Rank

Constructs		Valid	Spearman	t(N-2)	p-value
MisAlign	& Collab	46	0,55	4,33	0,00
MisAlign	& DKT	46	0,36	2,54	0,01
MisAlign	& TKD	46	0,29	2,00	0,05
MisAlign	& Pers	46	0,30	2,07	0,04
MisAlign	& Acc/Process	46	0,64	5,48	0,00

The table above illustrates that the correlation between the two variables highlighted in red are highly significant. There is a highly significant correlation between misalignment (MisAlign) and the following variables: poor collaboration (Collab), lack of developer knowledge of testing (DKT), conflicting interpersonal skills (Pers) and lack of accountability merged with process non-compliance (Acc/Process).

All these independent variables have a *p*-value < 0.05 hence they are discussed as having a significant correlation with misalignment. On the contrary, lack of tester knowledge of development is not seen as having a highly significant correlation with misalignment. The results of the Spearman's rank correlation are illustrated in table 15 below together with the associated hypotheses.

Table 15 Results of Spearman's Rank Correlation and Hypotheses

Dependent Variable	**Hypothesis**	**Independent Variable**	***p*-level**
Misalignment	H1	Lack of Communication	N/A
	H2	**Poor Collaboration (COLLAB)**	**0,000**
	H3a	**Lack of Developer Knowledge of Testing (DKT)**	**0,015**
	H3b	Lack of Tester Knowledge of Development (TKD)	0,051
	H4	**Conflicting Interpersonal Skills (PERS)**	**0,045**
	H5 and H6	**Process Non-compliance and Lack of Accountability (ACC/PROCESS)**	**0,000**

Note that hypothesis 1 (H1) which refers to lack of communication was not tested because this construct was not successfully validated during confirmatory factor analysis and was thus eliminated.

The results of the Spearman's rank correlation resulted in all the hypotheses being supported as illustrated in table 15. Thus, the hypotheses supported as per Spearman's rank correlation are outlined below:

H2 – poor collaboration and lack of communication positively influence misalignment

H3a – lack of developer knowledge of testing positively influences misalignment

H4 – conflicting interpersonal skills positively influence misalignment

H5 and H6 – process non-compliance and lack of accountability positively influence misalignment

4.2.8 Refined Conceptual Model

The results in the table 7 lead to the refined conceptual model illustrated below. The construct and discriminant validity resulted in several variables being removed and some were merged to form one variable. Lack of Communication (COMM) was removed from the independent variables while Process Non-compliance (PROCESS) and Lack of Accountability (ACC) were combined to form one variable denoted as ACC/PROCESS. The factor analysis resulted in a two-factor structure for shared domain knowledge (developer knowledge of testing and tester knowledge of development). The refined conceptual model is illustrated below in figure 3.

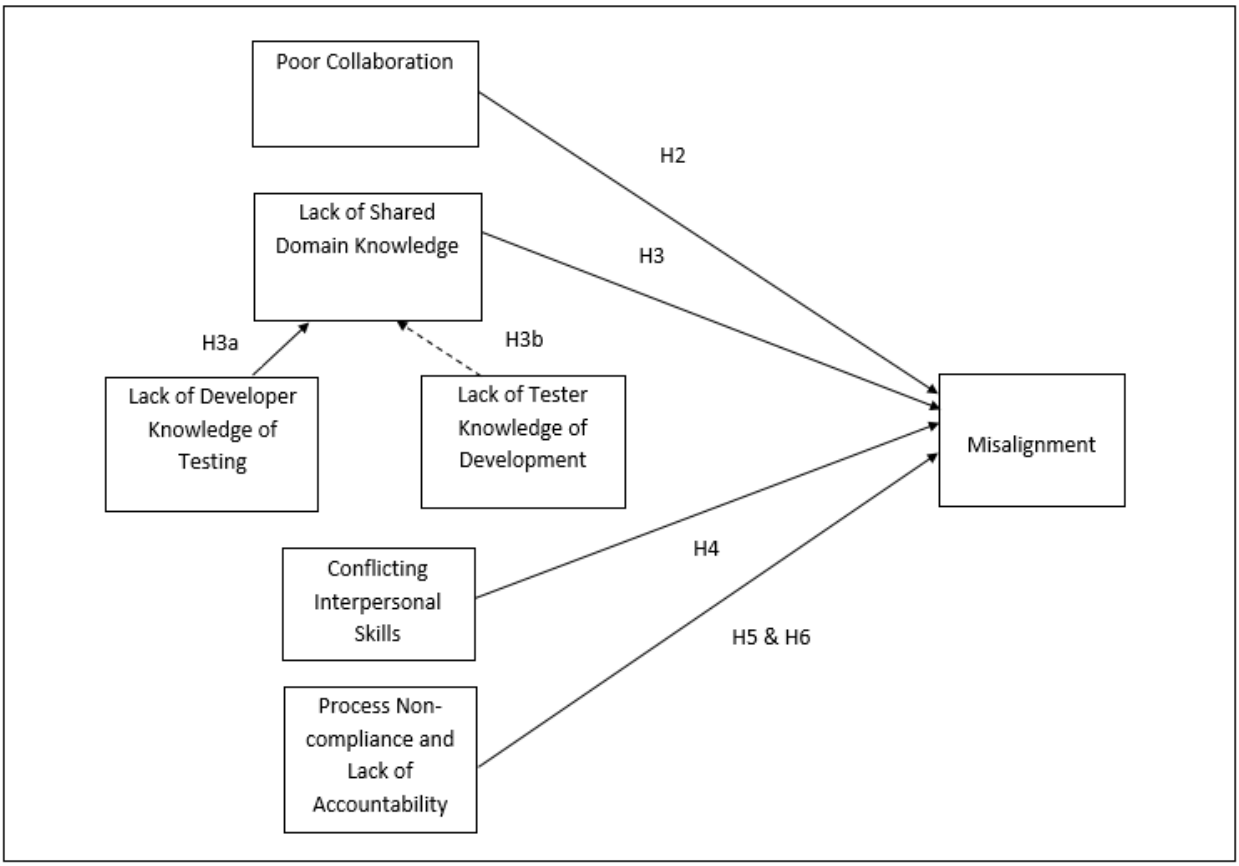

Figure 3 Refined Conceptual Model

5. CHAPTER FIVE: RESEARCH LIMITATIONS, CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH

5.1 Introduction

The following section includes the discussion and implications arising from the data analysis that was conducted in the previous chapter. Conclusions are also drawn from the results of the data analysis followed by research limitations, recommendations for future research and conclusion.

5.2 Discussion and Implications

According to literature, the following variables influence misalignment between developers and testers in agile software development companies:

- Lack of Communication (Guimaraes, Staples, & McKeen, 2004)
- Poor Collaboration (Mellin, et al., 2010)
- Lack of Shared Domain Knowledge (Espinosa, Slaughter, Kraut, & Herbsleb, 2007)
- Lack of Accountability (Wood & Winston, 2007)
- Conflicting Personalities (Alge, Gresham, Heneman, Fox, & McMasters, 2002)
- Process Non-compliance (Kohlbacher & Gruenwald, 2011)

Analysis was conducted on the data that was collected from developers and testers. The analysis was conducted through various statistical tools to test if the above factors do in fact influence the misalignment between developers.

The results of the data analysis reflected that most of the variables discussed above influence misalignment between developers and testers, however there was some difference from these factors based on the data collected and validation, which resulted in lack of shared domain knowledge being broken down to lack of developer knowledge of testing and lack of tester knowledge of development. Furthermore, after progressive refinement of the research instrument, process non-compliance and lack of accountability were merged into one variable, and lack of communication fell away due to its poor validity as a construct.

Poor collaboration positively influences misalignment between developers and testers. This suggests that where there is little or no interaction between developers and testers on the tasks that need to be done within the team, this has a negative impact on the overall software development project since both roles have a pivotal role to play in the team.

Lack of shared domain knowledge positively influences misalignment between developers and testers. The lack of shared domain knowledge is further broken down into lack of developer knowledge of testing and lack of tester knowledge of development. Lack of developer knowledge of testing is the key influence, while tester knowledge of development is not a major influence. This contradicts Dhaliwal, Onita, Poston, & Zhang (2011) who posit that testers need to have some form of technical expertise in order to thrive in software development projects, but supports the view that developers need to have good interpersonal skills if they want to work well with testers. This apparent contradiction can be explained by the fact that the social dimension of misalignment was not taken into consideration by Dhaliwal et al., (2011). Thus, if developers learn to exchange knowledge for the overall success of the project, they could work like a well oiled machine and possibly be more productive and efficient. Furthemore, conflicting interpersonal skills positively influence misalignment as well as process non-compliance and accountability combined.

Developers and testers have different personalities and capabilities. According to Faheem, Fernando, Salah, & Piers (2013) people's personality usually determine their capabilities. Developers are introverted in general and thus tend to possess technical skills while testers tend to be more extroverted in general and thus have better interpersonal skills than developers. Both parties need to be aware of these conflicting interpersonal skills and strive to meet each other half way in order to establish a good working relationship.

Previous studies such as Dhaliwal, Onita, Poston, & Zhang (2011) only highlight the intellectual dimensions of alignment while this study makes a contribution by empirically investigating the social dimensions which result in the misalignment between two prominent actors in software development projects. Dhaliwal, Onita, Poston, & Zhang (2011) address alignment between development and testing functions whilst this study looks at the social dimension of alignment by addressing alignment between the individuals in order to bring a new dimenstion to the study of alignment in IT organizations.

5.3 Contributions of This Research

The study contributes to the body of knowledge by illustrating a refined theoretical framework and demonstrating validity and reliability of the variables for measuring misalignment between developers and testers and factors of influence. The refined model illustrates that all the validated factors do in fact influence misalignment. In the refined model, lack of communication is absent. Lack of shared domain knowledge is broken down to lack of developer knowledge of testing and lack of tester knowledge of development. Process non-compliance and lack of accountability are merged and illustrated as one variable because the respondents provided similar responses for these variables. The results of the data analysis serve as evidence to support most of the hypotheses that were initially drawn up thus it can be concluded that all the said factors postively influence the misalignment between developers and testers.

The findings of the study can serve as a guideline for managers of software development projects in agile organizations to know the factors that influence misalignment between developers and testers. Once they have identified the causes of the problem, it will be easier for them to identify strategies to alleviate this problem. This in turn will result in a better working relationship between developers and testers and thus productivity will be improved when the relationship between the two parties is more aligned.

5.4 Limitations and Recommendations for Future Research

The primary objective of the study was to investigate the factors that influence misalignment between developers and testers in agile organizations. Through extensive literature review, the study uncovered 6 factors which influence misalignment between developers and testers. A suggestion for future research could be to identify other factors which influence misalignment between developers and testers in order to expand on the factors that were identified in this study. Furthermore, a better measure of lack of communication can be investigated since this measure fell away after the validity tests.

Agile organizations was the study's primary focus without taking into account traditional plan-driven organizations. Future research could try and investigate factors that influence misalignment in traditional plan-driven organizations and further compare if these factors differ to the one's identified in agile software organizations.

The study merely identified the factors which influence misalignment between the two parties but did not proceed to suggest strategies to alleviate this problem. Therefore, future research could expand on this study by identifying alleviation strategies that agile organizations can employ to address this problem.

Lastly, the data collection resulted in fewer responses than what was anticipated. The survey was sent out to approximately 100 respondents but only 46 valid responses were received. This resulted in the conclusions of this study being based on a small sample. Future research could try and produce a higher responses rate in order to test if all the hypotheses in the study will still be supported when the sample size is larger and this may resolve the overlap between lack of accountability (ACC) and process non-compliance (PROCESS).

5.5 Conclusion

The purpose of this study was to investigate the factors that influence misalignment between developers and testers in agile organizations. The research model included examining if the following factors positively influence misalignment: lack of communication, poor collaboration, lack of shared domain knowledge, conflicting interpersonal skills, process non-compliance and lack of accountability. Few studies have investigated misalignment between developers and testers in agile organizations. Most studies discuss lack of alignment between developers and testers in the context of traditional plan-driven organizations.

The findings confirm that misalignment between developers and testers in agile organizations is influenced by poor collaboration, lack of shared domain knowledge, conflicting interpersonal skills, process non-compliance and lack of accountability. Thus, these factors may have a negative impact on the overall success of software development projects in agile organizations if they are not addressed appropriately and subsequently alleviated.

6. REFERENCES

Alge, B. J., Gresham, M. T., Heneman, R. L., Fox, J., & McMasters, R. (2002). Measuring Customer Service Orientation Using A Measure Of Interpersonal Skills: A Preliminary Test In A Public Service Organization. *Journal of Business and Psychology, 16 (3)*, 467-476.

Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing.* New York: Cambridge University Press.

Arnicane, V. (2007). *Use of Non-IT Testers in Software Development.* Berlin: Springer.

Aversano, L., Grasso, C., & Tortorella, M. (2012). A literature review of Business/IT Alignment Strategies. *Procedia Technology, 5 (1)*, 462 – 474 .

Baron, R. A., & Tang, J. (2008). Entrepreneurs Social Skills and New Venture Performance: Mediating Mechanisms and Cultural Generality. *Journal of Management, 1(1)*, 1-25.

Beck, K. (2000). *eXtreme Programming Explained.* Boston: Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Kern, J. (2001). *The Agile Manifesto.* Ward Cunningham.

Biddix, J. P. (2014, May 01). *Word Press*. Retrieved from Word Press: http://researchrundowns.wordpress.com/quantitative-methods/instrument-validity-reliability/

Brown, I., & Jayakody, R. (2008). B2C E-commerce Success: A Test and Validation of a Revised Conceptual Model. *The Electronic Journal Information Systems Evaluation, 11(3)*, 167-184.

Brown, J., Lindgaard, G., & Biddle, R. (2011). Collaborative Events and Shared Artefacts:Agile Interaction Designers and Developers Working Toward Common Aims. *Agile Conference* (pp. 87-96). Canada: IEEE Computer Society.

Bruun, A., & Stage, J. (2012). Training Software Development Practitioners in Usability Testing: An Assessment Acceptance and Prioritization. *Proceedings of the 24th Australian Computer-Human Interaction Conference* (pp. 52-60). Australia: ACM.

Campbell, B., Kay, R., & Avison, D. (2005). Strategic Alignment: A Practitioner's Perspective. *Journal of Enterprise Information Management, 18 (6)*, 653-664.

Cohen, C., Birkin, S., Garfield, M., & Webb, H. (2004). Managing Conflict in Software Testing. *Communications of the ACM, 47 (1)*, 76-81.

Cohn, M., Leffingwell, D., Larman, C., & Vodde, B. (2013, January 15). *Scaled Agile Framework.* Retrieved from Scaled Agile Framework: http://scaledagileframework.com/developers-and-testers/

Crowther, D., & Lancaster, G. (2008). *Research Methods: A Concise Introduction to Research in Management and Business Consultancy.* United Kingdom: Butterworth-Heinemann.

Davis, R. (2014, December 22). *Rob Davis PE*. Retrieved from Rob Davis PE: http://www.robdavispe.com/free5/software-qa-testing-test-tester-2294.html

Dhaliwal, J., Onita, G. C., Poston, R., & Zhang, X. P. (2011). Alignment within the software development unit: Assessing structural and relational dimensions between developers and testers. *Journal of Strategic Information Systems, 20 (4)*, 323-342.

Espinosa, J. A., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems, 24 (1)*, 135-169.

Faheem, A., Fernando, C., Salah, B., & Piers, C. (2013). Soft Skills and Software Development: Reflection from the Software Industry. *International Journal of Information Processing and Management, 4 (3)*, 171-191.

Gandomani, T. J., Zulzalil, H., Ghani, A., & Sultan, A. B. (2012). Important Considerations For Agile Software Development Methods Governance. *Journal of Theoretical and Applied Information Technology, 55 (3)*, 345-351.

Greer, D., & Hamon, Y. (2011). Agile Software Development. *Software Practice and Experience, 41 (1)*, 943-944.

Guimaraes, T., Staples, D. S., & McKeen, J. D. (2004). Empirically Testing Some Main User-related Factors for Systems Development Quality. *Quality Control and Applied Statistics, 49 (1)*, 333-336.

Hanson, J. D., Melnyk, S. A., & Calantone, R. A. (2011). Defining and Measuring Alignment in Performance Management. *International Journal of Operations & Production Management, 31 (10)*, 1089-1114.

Hart, M., & Gabi, P. (2004). The Impact of Cognitive and Other Factors on the Perceived Usefulness of OLAP. *Journal of Computer Information Systems, 45 (1)*, 47-56.

Hauke, J., & Tomasz, K. (2011). Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data. *Quaestiones geographicae, 30 (2)*, 87-93.

Helquist, J. H., Deokar, A., Meservy, T., & Kruse, J. (2011). Dynamic Collaboration: Participant-Driven Agile Processes for Complex Tasks. *ACM SIGMIS Database, 42 (2)*, 95-115.

Joseph, A. M., & Kavita, M. (2008). *SAGE Publications.* Retrieved from SAGE Pub: http://srmo.sagepub.com/view/sage-encyc-qualitative-research-methods/n164.xml

Kettunen, V., Kasurinen, J., Taipale, O., & Smolander, K. (2010, July 12-16). A Study on Agility and Testing Processes in Software Organizations. Treno, Italy, Europe.

Kohlbacher, M., & Gruenwald, S. (2011). Process Orientation: Conceptualization and Measurement. *Business Process Management Journal, 17 (2)*, 267-283.

Lagestee, L. (2013, May 07). *Illustrated Agile*. Retrieved from Illustrated Agile: http://illustratedagile.com/2013/05/07/when-developers-and-testers-collide/

Levin, K. (2015, December 15). *Study design III: Cross-sectional Studies.* Scotland: University of Dundee. Retrieved from Institute for Work and Health: http://www.iwh.on.ca/wrmb/cross-sectional-vs-longitudinal-studies

Lewis, W. E. (2016). *Software Testing and Continuous Quality Improvement.* Florida: CRC Press.

Linders, B. (2013, May 23). *Info Q.* Retrieved from Improving Collaboration of Testers and Developers in Agile Teams: http://www.infoq.com/news/2013/05/collaboration-developer-tester

Littlejohn, S. W., & Foss, K. A. (2011). *Theories of Human Communication.* Aufl: Long Grove: Waveland.

Maharaj, S., & Brown, I. (2015). The Impact of Shared Domain Knowledge On Strategic Information Systems Planning and Alignment: Original Research. *South African Journal of Information Management, 17 (1)*, 1-12.

Marczak, S., & Damian, D. (2011). How Interaction Between Roles Shapes the Communication Structure in Requirements-Driven Collaboration. *19th International Requirements Engineering Conference* (pp. 47-56). Canada: IEEE.

Marczyk, G., DeMatteo, D., & Festinger, D. (2005). *Essentials of Research Design and Methodology.* New Jersey: John Wiley & Sons Inc.

McElfish, S. S. (2011). *Identifying Cultural Changes Necessary in Traditional Plan-Driven Software Development Organizations when Preparing to Adopt Agile Principles.* Eugene, USA: University of Oregon.

McHugh, O., Conboy, K., & Lang, M. (2012). Agile Practices: The Impact on Trust in Software Project Teams. *Software IEEE, 29 (3)*, 71-76.

Mellin, E. A., Bronstein, L., Anderson-Butcher, D., Amorose, A. J., Ball, A., & Green, J. (2010). Measuring Interprofessional Team Collaboration in Expanded School Mental Health: Model Refinement and Scale Development. *Journal of Interprofessional Care, 24 (5)*, 514-523.

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A Teamwork Model for Understanding an Agile Team: A Case study of a Scrum Project. *Information and Software Technology, 52 (5)*, 480-491.

Muller, C. (2013). *Body - Language - Communication.* Germany: Hubert & Co.

Myers, M. (1997, December 15). *Association for Information Systems*. Retrieved from Association for Information Systems: http://www.qual.auckland.ac.nz/

Nemoto, T., & Beglar, D. (2014). Likert-Scale Questionnaires. *JALT 2013 Conference Proceedings* (pp. 1-8). Tokyo: JALT.

O'Boyle, E., Humphrey, R., Jeffrey, Pollack, M., Hawver, T., & Story, P. (2011). The Relation between Emotional Intelligence and Job Performance: A Meta-analysis. *Journal of Organizational Behavior, 32 (5)*, 788-818.

Onita, C., & Dhaliwal, J. (2011). Alignment Within the Corporate IT Unit: An Analysis of Software Testing and Development. *European Journal of Information Systems, 20 (1)*, 48-68.

Parveen, T., Tilley, S., & Gonzalez, G. (2007). A Case Study in Test Management. *Proceedings of the 45th Annual Southeast Regional Conference* (pp. 82-87). North Calorina: ACM.

Pennington, R. R., & Kelton, A. S. (2016). How Much is Enough? An Investigation of Nonprofessional Investors Information Search and Stopping Rule Use. *International Journal of Accounting Information Systems, 21 (1)*, 47-62.

Reich, B. H., & Benbasat, I. (2000). Factors that Influence the Social Dimension of Alignment Between Business and Information Technology Objectives. *MIS Quarterly, 24 (1)*, 81-113.

Ritchie, J., Lewis, J., & Elam, G. (2003). Designing and Selecting Samples. *A Guide for Social Science Students and Researchers, 2(1)*, 111-145.

Robles, M. (2012). The Executive Perceptions of Top 10 Soft Skills Needed in Today's Workplace. *The Executive Perceptions of Top 10 Soft Skills Needed in Today's Workplace*, 453-465.

Sargunar, J. (2011). *Introduction to Computer Science.* India: Dorling Kindersley.

Saunders, M., Lewis, P., & Thornhill, A. (2011). *Research Methods for Business Students 5th Edition.* India: Pearson Education.

StatSoft. (2016, September 24). *About us: StatSoft*. Retrieved from StatSoft Web site: http://www.statsoft.com/Company/About-Us

Sumrell, M. (2007). From Waterfall to Agile – How Does a QA Team Transition? *AGILE Conference 2007* (pp. 291-295). Washington DC: IEEE.

Tan, M., & Teo, T. S. (2000). Factors Influencing the Adoption of Internet Banking. *Journal of the AIS, 1 (1)*, 5.

Tongco, M. D. (2007). Purposive Sampling as a Tool for Informant Selection. *A Journal of Plants, People and Applied Research*, 147-158.

Tripathi, V., & Goyal, A. K. (2014). Agile Testing Challenges and Critical Success Factors. *International Journal of Computer Science & Engineering Technology, 1(5)*, 632-638.

Trochim, W., Donnelly, J. P., & Arora, K. (2015). *Research Methods: The Essential Knowledge Base.* Canada: Nelson Education.

VanVoorhis, C. R., & Morgan, B. L. (2007). Understanding Power and Rules of Thumb for Determining Sample Sizes. *Tutorials in Quantitative Methods for Psychology, 3 (2)*, 43-50.

Vijayvargiya, A. (2009). One-way Analysis of Variance. *Journal of Validation Technology, 15(1)*, 62-63.

Waters, K. (2007, October 11). *All About Agile*. Retrieved from All About Agile: http://www.allaboutagile.com/how-to-implement-scrum-in-10-easy-steps-step-4-sprint-planning-tasks/

Weiss, C. (2008). *Commercial meets Open Source - Tuning Statistica with R.* Germany: University of Wurzburg.

Wood, J. A., & Winston, B. E. (2007). Development of Three Scales to Measure Leader Accountability. *Leadership & Organization Development Journal, 28 (2)*, 167-185.

Yu, B. L., Wooi, K. L., Wai, Y. T., & Soo, F. T. (2012). Software Development Life Cycle Agile vs Traditional Approaches. *2012 International Conference on Information and Network Technology* (pp. 162-167). Singapore: IACSIT Press.

Zhang, X., Stafford, T. F., Dhaliwal, J. S., Gillenson, M. L., & Moeller, G. (2014). Sources of Conflict between Developers and Testers in Software Development. *Information & Management, 51(1)*, 13-26.

7. APPENDICES

7.1 APPENDIX A: Ethics Approval Letter

UNIVERSITY OF CAPE TOWN

Faculty of Commerce

Ethics in Research Committee

University of Cape Town Private Bag Rondebosch 7701

Email: kincaidharold592@gmail.com

Telephone: 071 823 7573

February 16, 2015

Unathi Mbekela

Information Systems

Project title: Investigating the Factors that Influence Misalignment between Developers and Testers in Agile Organizations

Proposal no. 16-2015

Dear Researcher,

This letter serves to confirm that this project as described in your submitted protocol has been approved.

Please note that if you make any substantial change in your research procedure that could affect the experiences of the participants, you must submit a revised protocol to the Committee for approval.

Regards,

Professor Harold Kincaid

Signed by candidate

Signature Removed

Commerce Faculty Ethics in Research Committee

7.2 APPENDIX B: Cover Letter

Department of Information Systems
Leslie Commerce Building
Engineering Mall, Upper Campus
OR
Private Bag X3 - Rondebosch - 7701
Tel: +27 (0) 21 650 2261 Fax: +27 (0) 21650 2280
Internet: http://www.commerce.uct.ac.za/informationsystems/

03 March 2016

Dear Sir/Madam,

I am studying towards a Master of Commerce degree in Information Systems. As part of my studies I must complete a research paper. Your participation to this research will be greatly appreciated and will allow me to understand why there is a misalignment between developers and testers in agile organizations and how this problem can be alleviated. This research has been approved by the Commerce Faculty Ethics in Research Committee.

I kindly request you to take part in this research by completing the survey questionnaire in the following link <insert link>. Only your own opinion is important. You do not have to give your name, so no one will find out what your answers were. We cannot and do not want to find out who answered what.

The collective findings of this study will be captured in a report that will be presented to the University of Cape Town for academic purposes. The findings may also be published in an academic journal or presented at a conference if the information is deemed of academic value.

To complete this questionnaire should take no longer than **20 minutes**.

PREREQUISITE: To complete this survey questionnaire you must be a developer, tester or IT manager that works for an IT department/company and is involved in an IT/software development project.

DUE DATE: Survey questionnaire must be completed no later than <tbc>.

IMPORTANT: You do not have to complete this survey questionnaire. Participation is completely voluntary. You can choose to withdraw from the research at any time.

Should you have any questions regarding the research please feel free to contact the researcher Unathi Mbekela – 082 970 5502 or research supervisor Professor Irwin Brown - 021 650 3155.

Thank you for your time and cooperation.

Sincerely,

Unathi Mbekela

MComm Information Systems

Department of Information Systems

University of Cape Town

Email: umbekela@gmail.com

Professor Irwin Brown

Research Supervisor

Department of Information Systems

University of Cape Town

Email: Irwin.brown@uct.ac.za

ITJBrown

Signed by candidate

Signature Removed

7.3 APPENDIX C: Research Questionnaire

Study Purpose: To determine the factors which influence misalignment between developers and testers in agile organizations.

Prerequisite to complete survey:

- Are you a developer, tester or IT manager?
  - ❑ YES
  - ❑ NO
- Do you work for a software development company/department?
  - ❑ YES
  - ❑ NO
- Have you been/are you currently exposed to an agile software development project/s?
  - ❑ YES
  - ❑ NO

If you answered YES to all questions, please proceed to the next Section.

DEMOGRAPHIC INFORMATION

Please select **only one** option per question.

1. What is your current role?
   - ❑ Developer
   - ❑ Tester
   - ❑ IT Manager
   - ❑ Other

   If other, please specify _____
2. How long have you been in this role?
   - ❑ < 1 year
   - ❑ 1 - 3 years

- ❑ 3 – 5 years
- ❑ 5 – 7 years
- ❑ > 7 years

3. What is your level of management experience?
   - ❑ Junior
   - ❑ Intermediate
   - ❑ Senior
4. How much agile software development experience do you have?
   - ❑ < 1 year
   - ❑ 1 - 3 years
   - ❑ 3 – 5 years
   - ❑ 5 – 7 years
   - ❑ > 7 years
5. How many agile software development projects have you been involved in?
   - ❑ 1
   - ❑ 1 - 3
   - ❑ 3 – 5
   - ❑ 5 – 7
   - ❑ > 7

Please respond to the following questions with respect to your collective experience with agile software development project/s:

FACTORS WHICH INFLUENCE MISALIGNMENT					
Please circle the appropriate option for each statement.	**Strongly Agree**	**Agree**	**Not Sure**	**Disagree**	**Strongly Disagree**
Misalignment					
1. Developers and Testers do not have a shared understanding of agile software development projects	1	2	3	4	5
2. Developers and Testers do not adhere similarly to agile software development projects	1	2	3	4	5
3. Developers and Testers do not have a clear link between their combined efforts and outcome of a project	1	2	3	4	5
4. Developers and Testers do not collectively set standards and targets which they need to meet	1	2	3	4	5
Lack of Communication					
5. There is a lack of a common language between Developers and Testers	1	2	3	4	5
6. Developers and Testers do not listen to each other	1	2	3	4	5
7. Developers and Testers do not express their ideas clearly to each other	1	2	3	4	5
Poor Collaboration					
8. Interaction between Developers and Testers is minimal	1	2	3	4	5
9. Mutual respect between Developers and Testers is lacking	1	2	3	4	5

10. Co-operation between Developers and Testers is poor	1	2	3	4	5
11. Collective ownership of goals between Developers and Testers is lacking	1	2	3	4	5
Lack of Shared Domain Knowledge					
12. Developers lack knowledge on how to thoroughly test software	1	2	3	4	5
13. Developers lack knowledge on the value added by Testers in an agile software development team	1	2	3	4	5
14. Developers do not have strong soft skills	1	2	3	4	5
15. Testers lack knowledge on how to resolve code issues	1	2	3	4	5
16. Testers lack knowledge on the type of testing done by developers	1	2	3	4	5
17. Testers lack technical software development skills	1	2	3	4	5
Lack of Accountability					
18. Developers and Testers do not take joint responsibility for their actions in a team	1	2	3	4	5
19. There is lack of transparency between the work of Developers and Testers	1	2	3	4	5
20. There is a lack of openness between Developers and Testers	1	2	3	4	5
21. Developers and Testers fail to jointly account for the outcomes of their actions and decisions	1	2	3	4	5
Conflicting Personalities					
22. Testers tend to be more extroverted than developers	1	2	3	4	5

23. Testers tend to be more diplomatic than developers	1	2	3	4	5
24. Developers and Testers usually do not get along	1	2	3	4	5
Process Non-Compliance					
25. Agile software development processes are not clearly designed	1	2	3	4	5
26. Agile software development processes are not clearly documented	1	2	3	4	5
27. Process performance measurements are poorly conducted in agile software development teams	1	2	3	4	5
28. There is a lack of process improvement skills among Developers and Testers	1	2	3	4	5