Influencers of enhanced performance in Agile Software Development Teams

A

Research Report

By

Mmadira Njomo

(866821)

In

Partial fulfilment of the requirements for the

MASTERS IN COMMERCE (INFORMATION SYSTEMS)

(Course work)

At the

School of Economic and Business Sciences

University of the Witwatersrand

Supervisor:

Prof Ray Kekwaletswe

Abstract

Due to the trite nature of the software development environment, traditional software methodologies are no longer relied on to deliver software products in a timeous manner. As a response to this limitation, the agile manifesto was launched. The manifesto consists of values and principles centred around the self-organising team's ability to achieve higher productivity, that is, to deliver software products quickly and with a high quality.

With the self-organising team at the centre of this phenomenon, this interpretive case study seeks to gain greater insight into the processes and reasons behind this outcome. The site selected for this study is the IT divisions of a South African bank that have adopted Agile as a methodology to deliver software products.

The data was collected through semi structured interviews, focused groups and documentation. The data was analysed qualitatively using thematic and content analysis. The framework for enhanced performance in agile software development teams was conceptualised. The conceptualisation was informed by the empirical evidence and the interpretation of findings and literature

Keywords:

Agile software development; test-driven development, shared mental models theory, continuous delivery, self-organising teams

AKNOWLEDGEMENT

I would like to thank my supervisor Prof Ray Kekwaletswe for the support and guidance during the study and the delivery of the research report.

To ABC bank (pseudonym) for providing access to your company information as well as interview participants.

To my husband, Fanna Njomo, for providing me with unwavering support and encouragement, thank you.

Mmadira E Njomo

…….
15 June 2017

Contents

Abstract 1
AKNOWLEDGEMENT 2
Contents 3
Chapter One (1): Introduction and Background 5
1.1. Introduction to the field of study 5
1.2. Background to the research problem 7
1.3. Problem Statement 8
1.4. The objectives of the study 9
1.5. Research Question 9
1.5.1. Secondary questions 9
1.6. Summary of Chapter 10
Chapter Two (2): Survey of Scholarship and Theoretical Framework 11
2.1. Introduction 11
2.2. Software development 11
2.2.1. Agility in Software Development 12
2.2.2. The Agile Manifesto 12
2.2.3. Agile Principles 13
2.2.4. Defining agile software development 14
2.3. Agile Software Development Methods 16
2.3.1. Research on Agile Methods 16
2.4. SCRUM 19
2.4.1. The Sprint 20
2.4.2. Scrum Roles 21
2.5. Software Development Teams 24
2.5.1. Agile Software Development Teams 25
2.5.2. Self-Organisation 25
2.5.3. Teamwork 27
2.6. Theoretical Frameworks 27
2.6.1. Dickinson and McIntyre Model 28
2.7. Conceptual Research Framework 31
2.8. Summary of Chapter 33
Chapter Three (3): Research Methodology 34
3.1. Introduction 34

3.2. Research Paradigm 34
3.2.1. Positivism 34
3.2.2. Critical Philosophy 35
3.2.3. Interpretivist Paradigm 36
3.3. Research Design and Strategy 37
3.3.1. Research Design 37
3.3.2. Research Strategy 37
3.4. Case Study 37
3.5. Case Description (Study location and context) 38
3.6. Data Collection 43
3.7. Ethics Consideration 46
3.8. Limitations of the study 46
3.9. Summary of Chapter 3 47
Chapter Four (4): Data Analysis and Discussion of Findings 48
4.1 Introduction 48
4.2 Thematic Analysis 48
4.3 Analysis of data collected 50
4.4 Participant demographic 50
4.5 Analysis of qualitative data collected 50
4.6 Summary of Chapter 56
Chapter Five (5): Interpretation of Findings 57
5.1 Introduction 57
5.2 Interpretation of findings 57
5.3 Summary of chapter 66
Chapter Six (6): Conclusion and Recommendation 67
6.1 Introduction 67
6.2 Implications of the study 67
6.3 Recommendation of the study 68
6.4 Conclusion 69
6.5 Limitations of the study and future research. 69
References 70
a. Appendix A. 72

Chapter One (1): Introduction and Background

1.1. *Introduction to the field of study*

Agility as an approach to software development was brought into the mainstream through the launch of an Agile Manifesto in 2001 by a group of 17 industry leaders in software development, this in response to a growing dissatisfaction with productivity levels of prevailing traditional software development methods (Fowler and Highsmith, 2001). The traditional software development methods referred hereto are identified as system development lifecycle methods (SDLC); characterised by a waterfall approach to software development; an approach that is plan driven, process based and calls for a rigid adherence to a sequence of pre-defined steps in the delivery of software products.

The proponents of the Agile manifesto attribute the dissatisfaction with the waterfall method to the notion that the approach inhibits flexibility in a world plagued by a rapid pace in information technology change (Boehm, 2002).

In contrast, Agile software development is characterised by flexibility and responsiveness to the inevitability of changing business requirements, this is seen as more productive than the waterfall approach. The difference between Agile and Waterfall approaches is that in an Agile approach: *interactions and individuals are valued more than processes and tools; working software is valued more than comprehensive documentation; Customer collaboration over comprehensive documentation and finally, responsiveness to change is valued more than adherence to a plan* (Beck et al, 2001).

The 2016 Standish Group Chaos report puts the overall success rate of software development projects at 16%, a successful project is defined by the Standish as one that is delivered on time, within budget and with agreed scope. The rest of the projects were either challenged; that is, they we delivered late, overshot their budget

and with less than agreed scope *or* they were cancelled altogether. While the group doesn't make any distinction of the methodology used that led to either success or failure, they do however offer some of the factors that increases the chances for success; these are listed as smaller timeframes, iterative development cycles, early user involvement and a clear and precise statement and set of objectives (Chaos Report, 2016). Although the Standish group did not attribute the success of 16% of the projects to Agile Software development per se, the listed recommendations are synonymous with Agile principles, at least at face value.

The introduction of agile as an alternative software development approach has piqued the interest of both practitioners and researchers alike, resulting in significant strides being made in the advancement of the field through research. A recent survey of the most researched area of Agile approaches to software development has found that interest in research still lies largely in Agile adoption and Critical Success Factors (Chuang, Loura and Lu, 2014).

When one takes a closer look at the studies, one can identify that there is a common denominator across all these studies on CSF, the presence of a self-organising team seems to be the hallmark of Agile software development, regardless of the Agile practice chosen (Hoda, Noble and Marshall, 2013). However, despite identifying the self-organising team as a critical to the success to the Agile approach to software development, there is still a paucity of studies explaining how this self-organising phenomenon manifests itself into producing the productivity levels enjoyed by the agile software teams.

It is, therefore, the intention of this study to explain and describe how a self-organising team produces enhanced levels of performance. Enhanced performance is herein defined as delivery of projects on time, within budget and with full scope, (Standish Group, 2016). Although the overall success rate of Software development projects in general is still a very low 16% as mentioned above, it has been reported that a significant portion of that success rate is attributed to software development projects that adopted the Agile approach (*ibid*., 2016).

1.2. *Background to the research problem*

Despite the self-organising team taking centre stage in the agile software development's growing popularity, there is a paucity of research on the phenomenon within software engineering (Hoda etal, 2013). This is viewed as a gap given that the above mentioned studies identify the most critical success factors of agile as the self-organising team, (Chow and Cao, 2008). If the Agile software methodology is to advance as a discipline, more studies of agile teams and the attributes these teams possess that enable them to produce enhanced levels of productivity need to be undertaken. It is not enough to study adoption and success factors if the core element that brings it all together, in this case the agile team, is not empirically tested.

Understanding and exploring what it takes for an agile team to be more productive could explain why some agile teams fail to produce enhanced productivity and others succeed. It is important to realise that not every team that claims to practice Agile is/ will be successful. A study of this nature could be helpful in identifying which attributes to develop and which to ignore when a decision to either adopt or continue with agile practises is being made.

Anecdotal evidence suggests that South African organisations have also joined in the discourse and that Agile as an approach to software development has been embraced and adopted by some of these organisations. However, there are insufficient studies undertaken in the South African context that would provide more relevant guidelines over and above what exists in the mainstream research. The South African geo-social environments make the case a unique one in the sense that it may uncover unexpected influencers. This is supported by the fact that a search on Self organising teams in Agile software development in the South African Journal publications elicits a negligible number of studies undertaken on this topic.

It is the intention of this study to advance the field of Agile Software development by providing the South African context into the discourse. The unit of analysis in this study is *teams*, specifically agile software development teams within the context of one of South Africa's four large banks. The bank in question has chosen to participate anonymously in this study and would therefore be referred to by the pseudonym, *ABC Bank*.

ABC Bank has adopted Agile Software development as a new approach to delivering software projects. The Agile practice adopted as most suitable by the bank is SCRUM. The objectives that ABC Bank sought to achieve with the adoption of this approach are listed as follows on their intranet website:

- ✓ Greater levels of certainty to deliver results
- ✓ Quick time to market
- ✓ Ability to respond to changes quickly
- ✓ Improved communication
- ✓ Skilled and flexible teams
- ✓ Increased employee satisfaction
- ✓ Opportunities to create/ exploit competitive advantage
- ✓

ABC Bank has set up their project teams in line with SCRUM with the expectation that the teams will enable the organisation to deliver on the above mentioned objectives. ABC Bank offers a suitable environment for the enquiry into the attributes that enable agile software development teams to enhance their levels of productivity as this aligns to the intentions of the Agile Manifesto.

1.3. ***Problem Statement***

The success and failures of agile software development practises are attributed to the team's ability or inability to self- organise. However, there is a paucity of studies dedicated to such teams. It is apparent that self-organising teams possess certain qualities or attributes that sets it apart and enables it to become more productive than traditional software development teams, what these qualities are and how they are acquired or utilised by agile teams has not been empirically determined.

The purpose of this research therefore, is to describe and explain the attributes that self–organising agile software development teams possess which provides them with the capability to enhance and sustain higher productivity levels.

1.4. *The objectives of the study*

This study sought the following:

- To describe how Agile Software development teams are formed
- To analyse the interpersonal skills that these team's posses
- To describe how the interpersonal skills may affect performance
- To describe factors that may enable or inhibit team performance.

1.5. *Research Question*

The research question addressed by this study was:

How do Agile Software development teams achieve enhanced levels of performance?

1.5.1. Secondary questions

- How do Agile Software development teams get formed?
- What are the skills that these teams possess?
- How does the presence or absence of interpersonal skills affect performance?
- What are the factors which influence the team's performance?

The rest of the research report is organised as follows:

- Chapter 2 of this report covers a detailed review of literature on agility in software development, Agile software development principles and practices, the self-organising agile software development teams.
- Chapter 3 explain the research methodology undertaken for the study as well a discussion on the teamwork model.
- Chapter 4 discusses the research findings and conclusions
- Chapter 5 presents opportunities for future research

1.6. *Summary of Chapter*

In this chapter we provided a background to the field of agile software development by discussing the origin of agile software development as it is currently understood, the reasons for its relevance as well as some of the challenges identified within the field. We have learned that at the centre of the success of agile software development lies agile team itself. A paucity of studies into what makes the agile team more productive than a traditional software team was stated as problem statement. The research question as well as secondary research questions were stated and discussed. In the next chapter we will perform an in-depth review of the literature on agile.

Chapter Two (2): Survey of Scholarship and Theoretical Framework

2.1. *Introduction*

In this chapter we review published literature on Agile software development in general and agile software development teams in particular. The rationale behind agility in software development, the strides that have been made as well as how different schools of thoughts define agile. We will also review the literature on the different agile practises available to any organisation to be adopted, how an organisation can go about choosing the correct agile practice for themselves. The literature review will also include studies that focus specifically on agile development team, how they self -organise or self-manage and any factors that pertain to the success of failure of agile teams.

2.2. *Software development*

The development or delivery of software products may be categorised into two approaches, traditional and agile software development methodologies. Traditional software methods were developed first and have been applied to software development projects for over three decades with mixed results (Cohen, Lindvall and Costa, 2004). The traditional software development methods referred hereto are identified as system development lifecycle methods (SDLC); characterised by a waterfall approach to software development; an approach that is plan driven, process based and calls for a rigid adherence to a sequence of pre-defined steps in the delivery of software products

Due to the trite nature of the software development environment, traditional software methodologies could no longer be relied on to deliver software products in a timeous manner. As a result of this growing dissatisfaction, a new, more agile approach is being increasingly utilised as an alternative approach to the delivery of software products.

2.2.1. Agility in Software Development

Agility as an approach to software development was brought to light through the launch of an Agile Manifesto in 2001 by a group of 17 industry leaders in software development. These developers were responding to a growing dissatisfaction with productivity levels of prevailing traditional software development methods (Fowler and Highsmith, 2001). The proponents of the Agile manifesto attribute the dissatisfaction with the waterfall method to the notion that the approach inhibits flexibility in a world plagued by a rapid pace in information technology changes, therefore agile methods have been introduced to overcome these challenges, (Boehm, 2002).

Agility in software development means to strip away as much of the heaviness commonly associated with the traditional software development methodologies in order to promote faster responses to changing environments and user requirements, (Dyaba and Dingsoyr, 2008).

2.2.2. The Agile Manifesto

According to the Agile Alliance (2001), the purpose of the Agile Manifesto is to uncover better ways of delivering software by doing it and helping others do it. The ability to develop better software, according to the Agile manifesto, may be achieved by an adherence to agile values, principles and practices. These three aspects of Agile will be listed verbatim for the purpose of providing the user with a comprehensive understanding and appreciation of the objective of the agile manifesto. The agile values that support its purpose are identified by the Agile manifesto as:

- valuing individuals and interactions **over** processes and tools;
- working software over comprehensive documentation,
- customer collaboration **over** contract negotiation, and
- an ability to respond to change **over** following a plan,

While the items on the right hand side as deemed as important (processes, tools, document etc,) the items on the left should be valued more. The abovementioned values emanate from the alliance members' practical experiences of success and failures in software delivery endeavours, and the subsequent acceptance without question of the above values by the development community. Although there is a paucity of empirical evidence regarding the values mentioned above, anecdotal evidence seems to support that the items on the left do indeed produce better working software when compared to valuing items on the left more.

2.2.3. Agile Principles

In addition to the above mentioned values, The Agile alliance has also agreed on a set of 12 principles that would serve to inform the spirit with which agile practitioners are to adopt the methodology. The twelve principles of agile are listed below:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple weeks, to a couple of months with a preference to shorter timescale
4. Business people and developers work together daily throughout the project.
5. Build projects around motivated individuals, give them the environment and support they need and trust them to get the job done.
6. The most efficient and effective method of conveying information with and within a development team is face to face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity- the art of maximising the amount of work not done- is essential

11. The best architectures, requirements and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

While the agile alliance believes that a set of common purposes and principles will benefit the users of agile methodologies, variety and diversity of practises are still strongly advocated (Agile Manifesto, 2001). These methods or practises may be adopted and tailored to fit the needs of the organisation or to solve a specific software delivery problem (Highsmith, 2002).

While the manifesto has been widely adopted unconditionally in some quarters, it has been criticized in other quarters as being too vague, placing emphasis on working software instead of usable software (Laanti, Simila and Abrahamson 2013).

2.2.4. Defining agile software development

As seen in the previous sections, the concept of agility in software development is significantly broad, it is subject to multiple definitions and is open to different interpretations. In their 2013 study, Laanti, *et al* identified and discussed at least 10 definitions of agile software development methodology in Table 1, below. While there are certain common elements across all the different definitions, the focus appears to vary significantly. It is therefore imperative that a clear definition of what agile means in the context of this study is made.

After reviewing the ten definitions outlined therein, the definition most aligned with the essence of this study is one by Schuh (2004) wherein Agile is defined as '*building software by empowering and trusting people, acknowledging change as a norm, and promoting constant feedback'*.

The emphasis on the people aspect of agile definition is aligned with Boehm and Turner (2012)'s view that the most critical success factor in agile software development was more likely to be in the realm of people factors, this is the fundamental difference between agile software development and traditional software

development. The people-centric nature of this approach is perceived to make any transition to this agile challenging, (Gandomani and Nafchi, 2016).

Table 1: Definitions of agile software development [adapted from Kettunen 2009,

Source	Definition
Cockburn 2001	Being effective and manoeuvrable. Use of light but sufficient rules of project behaviour and the use of human and communication oriented rules
Anderson 2003	Rapid and flexible response to change
Schuh 2004	Building software by empowering and trusting people. Acknowledging change as a norm, and promoting constant feedback, producing more valuable functionality faster.
Lyytinen 2006	Discovery and adoption of multiple types of Information Systems Development innovations through garnering and utilizing agile sensing And response capabilities.
Subramaniam 2005	Uses feedback to make constant adjustments in a highly collaborative environment
Ambler 2007	Iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by selforganizing teams with “just enough” ceremony that produces high-quality software in a cost-effective and timely manner which meets the Changing needs of its stakeholders.
Nerur and Balijepally 2007	Evolutionary delivery through short iterative cycles, intense collaboration, self organizing teams and high degree of Developer discretion. Learning, teamwork, self-organization and Personal empowerment. Responsiveness And flexibility. Interchangeability of roles and jobs based on autonomy
IEEE 2007	Capability to accommodate uncertain or changing needs up to a late stage of the development (until the start of the last iterative development Cycle of the release).
Wikipedia 2007	Conceptual framework for software engineering that promotes development iterations throughout the lifecycle of the project

The next section discusses the different agile practises and practises.

2.3. *Agile Software Development Methods*

Agile software development is a broad term used to describe software development methods or practices that adhere to a set of values and principles as defined in the Agile Manifesto (Beedle et al., 2001). These methodologies have been created by experienced practitioners and constitute a set of practices for software development teams (Dyba and Dingsoyr, 2008). These practises are characterised by short, iterative cycles of development, driven by product features, periods of reflection and introspection; collaborative decision making, incorporation of rapid feedback, and continuous integration of code changes into the system under development, (Highsmith, 2002).

An organisation that seeks to adopt an agile software development method will numerous alternatives to select from. Adopting any of these methods will have a significant impact on the daily work of the teams within the organisation. Application of an Agile method ‘straight out of the box’ may fail in various ways (Krutchen, 2011). To make this transition easier and to yield quick wins, a recommendation made by Cohen et al (2004) is that an organisation should examine their current development processes, identify problematic areas and identify the techniques that addresses those specific areas, and gradually adopt more, based on the outcome of the chosen technique until such time as full transition to an agile methodology has been made.

2.3.1. Research on Agile Methods

A summary of the most commonly used Agile software development practises is presented below. This summary provides a brief overview of each methodology’s most salient techniques. Research on the below mentioned practises has gathered some momentum through the years with some practises getting more research attention than others. For instance, XP has received the most research coverage than all other ‘common’ agile practises (Dyaba and Digsoyr, 2008).

Table 2: Description of main agile methods (Dyba and Dingsoyr, 2008)

Agile Practice	*Focus*
Crystal Methodologies	A family of methods for co-located teams of different sizes and criticality: Clear, Yellow, Orange, Red, Blue. The most agile method, Crystal Clear, focuses on communication in small teams developing software that is not life-critical. Clear development has seven characteristics: frequent delivery, reflective improvement, osmotic communication, personal safety, focus, easy access to expert users, and requirements for the technical environment
Dynamic Software development method	Divides projects in three phases: pre-project, project life-cycle, and post project. Nine principles underlie DSDM: user involvement, empowering the project team, frequent delivery, addressing current business needs, iterative and incremental development, allow for reversing changes, high-level scope being fixed before project starts, testing throughout the lifecycle, and efficient and effective communication
Feature Driven Development	Combines model-driven and agile development with emphasis on initial object model, division of work in features, and iterative design for each feature. Claims to be suitable for the development of critical systems. An iteration of a feature consists of two phases: design and development
Lean Software development	An adaptation of principles from lean production and, in particular, the Toyota production system to software development. Consists of seven principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity, and see the whole
SCRUM	Focuses on project management in situations where it is difficult to plan ahead, with mechanisms for ‘‘empirical process control”; where feedback loops constitute the core element. Software is developed by a self-organizing team in increments (called ‘‘sprints”), starting with planning and ending with a review. Features to be implemented in the system are registered in a backlog. Then, the product owner decides which backlog items should be developed in the following sprint. Team members coordinate their work in a daily stand-up meeting. One team member, the scrum master, is in charge of solving problems that stop the team from working effectively
Extreme Programming 1 & 2	Focuses on best practice for development. Consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-h week, on-site customers, and coding standards. The revised ‘‘XP2” consists of the following ‘‘primary practices”: sit together, whole team, informative workspace, energized work, pair programming, stories, weekly cycle, quarterly cycle, slack, 10-minute build, continuous integration, test-first programming, and incremental design. There are also 11 ‘‘corollary practices

A maturity assessment of agile practises in respect to theory and research undertaken by Dyaba and Dingoyr, 2008 found the nature of research on agile still

nascent, with eXtreme Programming being the only discipline in the intermediate stage.

A recommendation was made that Scrum become the focus of future research, due to the fact that its adoption is gaining traction in practice, PriceWaterhouseCoopers, (2012) reports that 52% of organisations that adopt the agile approach to software development apply the Scrum methodology, as per figure below. The fact that Scrum comprises project management as part of its practises could be the reason behind the its accelerated level of adoption, especially for organisations that were entrenched in the traditional software development methods for the longest of time (Cristal, Wildt and Prikladnicki, 2008)

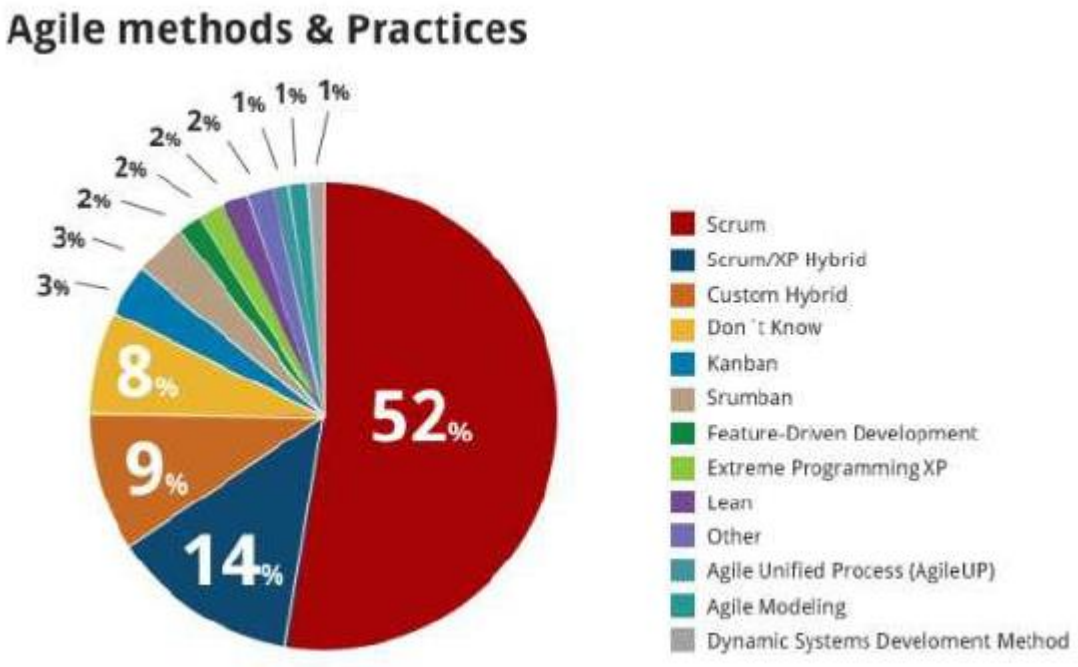


Figure 1: Assisting in the use of Agile methods (PriceWaterhouseCooper, 2012;)

The pace of agile adoption in practise has not been matched by a corresponding uptake in the research field. Significant gaps still remain in terms of theory and empirical research (Dyba and Dingsoyr, 2008). Together with many other studies (e.g., Dyba and Dingsoyr, 2008), this study has also heeded the call to narrow the gap by conducting a study to describe how agile software development teams, applying Scrum to achieve enhanced levels of productivity.

The next section will describe the Scrum process in more detail.

2.4. *SCRUM*

Scrum was inspired by inspired by product development teams in Japan, by Takeuchi and Nonaka where a small group of people were given the autonomy to go the distance and deliver on the goal, the concept was borrowed from the sport of rugby where a team goes the distance as unit, passing the ball back and forth till they reach their goal.

Scrum as an agile software development framework was developed by Ken Schwaber and Jeff Sutherland in 1995, and is defined as a framework within which people can address complex adaptive problems while productively and creatively delivering products of the highest value. Schwaber (1996) describes it as a process that appreciates the unpredictable nature of the software development process. The process itself is designed to be lightweight, easy to comprehend but difficult to master (Sutherland and Schwaber, 2016). Further descriptions of Scrum can be found throughout the literature as more and more studies on the topic are published.

The objective of the Scrum methodology is to deliver as much quality software as possible within a short period of time (Beedle, Devos, Sharon, Schwaber and Sutherland, 1998). Scrum allows the development of software in an iterative manner. Due to this iterative approach, it is not imperative that full requirements are written up front. This is also due to the belief that users and developers alike do not necessarily know what is possible upfront, at least not before the built process is underway.
Scrum was designed to achieve a hyper productive state where productivity increases by an order of magnitude over industry averages. Many small, collocated teams have achieved this effect (Sutherland et al, 2007).

According to the Scrum Guide by Sutherland and Schwaber (2016), Scrum consists of the following components:

- Scrum Team, which comprises of the roles, that of the product owner, development team and the Scrum master.

- Sprints, or Events, which comprise of the Sprint, sprint planning, daily scrum, sprint review, sprint retrospective
- Scrum Artefacts which comprise of product backlog, sprint backlog, increment.

The different scrum components will be discussed in detail in the following section.

2.4.1. The Sprint

A sprint is considered the heart of scrum, it is a time-box event of one month or less during which a complete, or 'Done' useable product or 'increment' is released, (Sutherland and Schwaber, 2016).

The following events take place within a sprint:

- ***Sprint planning*** is the event that takes place for the purpose of planning the work to be done within a sprint, called the sprint backlog. This backlog is created collaboratively by the entire team. According to the Scrum guide, the sprint planning session should be time-boxed to 8 hours, for a sprint that would last at most 1 month.
- ***Daily scrum meetings*** are a 15-minute event that takes place prior to the commencement of each workday. It is supposed to be held at the same place at the same time daily, its purpose is to enable synchronisation among team members. It maximises communication and collaboration among team members.
- ***Sprint reviews*** take place at the end of each sprint I order to review the outcome of the sprint, called an increment and update the product backlog. The sprint review can also be referred to as a showcase where the team, together with the product owner can interact with the product or increment that was developed and provide their feedback. According to the Scrum guide, the Sprint retrospective is time-boxed to 4 hours.
- Sprint retrospective also takes place at the end of the spring, preferably after the feedback from the review, where the scrum team reflects on the lesson's learned, what to continue to do and what needs to be adopted going forward.

The sprint can be referred to as the execution element of the scrum framework. While these events are meant to be prescriptive, specifically relating to the time-boxes, it is not immediately clear how the decisions on the lengths of the prescribed durations, (time-boxes) were informed. However, it is not within the scope of this study to make that determination.

2.4.2. Scrum Roles

A great Scrum Team consists of a Product Owner who maximizes value, a Scrum Master who enables continuous improvement and a Development Team who focus on delivering high quality product increments (Schwaeber, 2004).

The different scrum roles are identified in the revised Scrum Guide by Sutherland, and Schwaber (2016) as follows:

The first role in Scrum is that of the ***Product Owner***. This role is regarded as the most important as it is responsible for setting the objectives of the project and the funding thereof. The product owner is also responsible for maintaining and prioritising the so-called product backlog for the product being developed. The product backlog must be under constant care and scrutiny of the product owner in order to protect its integrity. This focus is the back bone of the activities of the product owner in his/ her collaboration with the scrum master, the team and the organisation's stakeholders. Together, they define the function of product (Pichler, 2010).

The responsibility of maximizing the value of the product and the work of the development team also lies with the product owner. It's a single -person role that brings the customer perspective of the product to a Scrum team. The product owner may be a representative of a committee, they may also choose to delegate any of their responsibility to other person or party, but they remain ultimately accountable. Furthermore, the product owner is also expected to:

- Develop and maintain a product vision and market strategy;
- Product management;

- Ordering and managing the Product Backlog;
- Involving stakeholders and end-users in Product Backlog refinement and backlog management;
- Alignment with other Product Owners when needed from an overall product, company or customer perspective

Secondly, the role of the ***Scrum Master*** is that of management and supervision of the project. The service they provide is three fold, to the product owner, the development team and the organisation at large.

Their service to the product owner is to provide the product owner with ways to optimise the product backlog management, continuously improving the team's understanding and appreciation of the product and sprint backlog and guiding the product owner to become more and more agile.

Their service towards the team is that of a coach, encouraging self-organisation, removing obstacles, instilling the discipline of producing high value/ quality products as well as setting up and facilitating scrum events when needed. It is also the responsibility of the scrum master to ensure that the scrum process is followed by the team. They are responsible for ensuring that Scrum is understood and enacted. This is done by ensuring that the Scrum team adheres to the Scrum theory, practices, and rules. Furthermore, the scrum master is the servant leader of the team. A great Scrum Master knows when and how to apply the scrum processes, depending on situation and context. No effort should be spared in helping people to get better at understanding and applying the Scrum framework. The scrum master also protects the team from external influence or contamination, helping those outside the Scrum team understands which of their interactions with the Scrum team are helpful and which are not (Overeem, 2016).

Their service to the organisation is to lead the organisation in the adoption of Scrum if need be, planning and scheduling scrum implementations within the organisation, creating understanding and providing clarity and teaming up with other scrum

masters to increase the effectiveness of agile in the organisation, (Sutherland and Schwaber, 2016; Overeem, 2016).

Lastly, the role of **Development Team** is fulfilled by people from various professional backgrounds who possess the required skills to execute on the sprint backlog, (Sverisdottir et al, 2014). It is recommended that the number of team members be between 3 and 9 team members excluding the product owner and scrum master. A team of too few people, in this case less than 3, may experience a capacity constrain while a team that is too large, greater than 9 team members may be challenging to coordinate. It is not immediately apparent whether these minimum and maximum number of team size is underpinned by any theory or empirically determined. However the determination of optimal team size falls outside the scope of this study.

In order to reach their goals, the development team need, for example, to understand and choose the right models and techniques to support their projects, consider key questions such choosing the best lifecycle model, establishing an appropriate balance of effort between documenting the work and getting the product implemented, the trade-off between spending major efforts on planning in advance and avoiding change, and when is it more beneficial to plan less rigorously and embrace change, (Cohen et al, 2004).

Scrum puts great emphasis on product control, flexibility and empowering teams to deliver on the product backlog (Sverridottir, 2014). It offers a framework that facilitates the team's learning through discovery, collaboration and experimentation.

Since Agile is very people focussed, there is a further need to understand how team size affects individual behaviour and productivity within an Agile team (Lalsing, Kishnah and Pudaruth, 2012). Figure 2 shows a graphical depiction of scrum. The concept of teams is discussed in the next section.

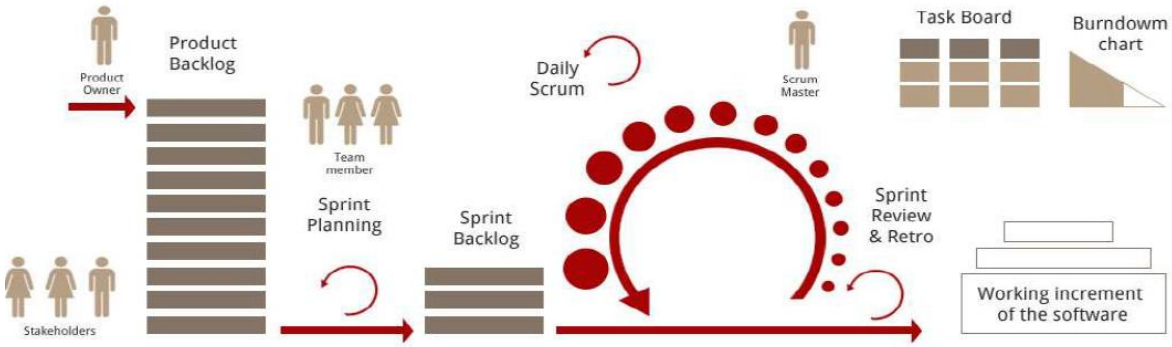


Figure 2: SCRUM Framework: Sverrisdottir et al, 2014

Scrum development teams need to structured and empowered by the organization to organize and manage their own work. Self-management is a defining characteristic of a scrum development team (Moe et al, 2009). Performance of agile teams is therefore enhanced by putting much more emphasis on teamwork, cooperation and self-organisation. (Moe et al 2008). The next section, in the line with the purpose of this study, will explore the self-organising team in more depth.

2.5. ***Software Development Teams***

Software development is described by Dingsoyr and Dyba (2012) as a socio-technical systems that consist of humans and technical entities that function in an integrated, coordinated manner in order to address a wide range of problems too complex to be addressed by either entity on its own. These problems pertain to the design and development of software. In the said study, they argue that the development of software tends to place more emphasis on the technological consideration than the socio component. It is the purpose of this study to attempt to close that gap by undertaking to put more emphasis on the socio, or the human aspects of software development.

Because software is mostly developed in teams (Dingsoyr and Dyba 2012), the focus of this enquiry is on the team itself. As mentioned in the preceding sections, the success of a software development project is significantly dependent on the role that its team members play. Thus it is imperative that the project environment allow

emergent team capabilities to manifest themselves in a mutually trustworthy and supportive atmosphere (Hastie, 2004).

2.5.1. Agile Software Development Teams

Agile software development teams are characterised by a small number of team members, usually not more than 9 members, who typically include someone that represents the customer, business analyst, developers and testers. These teams are capable of organising themselves, managing their own workload, allocating and reallocating work among themselves depending on need and best fit, they also make their own decisions as a team.

Takeuchi and Nonaka (1985), describe these teams as self-organizing teams, exhibiting autonomy, cross-fertilization, and self-transcendence. While much has been written on Agile software development teams generally, not nearly enough has been written about self-organising agile teams, even though agile teams are regarded as the Hallmark of agile software development (Hoda, Noble and Marshall; 2013)

2.5.2. Self-Organisation

Self-organizing teams took centre stage in the software engineering arena when they were incorporated as a hallmark of Agile software development. Self-organizing teams are not only seen as enabling Agile development practices, but also as capturing the spirit of Agile values and principles, which focus on human and social aspects (Hoda et al, 2013). An effective application of agile practises is dependent on the development team's ability to self-organise (Moe et al, 2008). However, a self-managing or self-organising team's performance depends not only on the team's competence in managing and executing its own work but also the organizational context provided by the management (Moe et al, 2008).

Although most studies report positive effects from self-managing teams, some present a more mixed assessment; self-organisation can be difficult to implement

and risk failure when used in inappropriate situations or without sufficient leadership and support. In their 2005 study, Salas, Sims and Burke, 2005 contend that the creation of a team of skilled members don't just happen, and that success is not always guaranteed.

What then is required for a team to successfully become self-managing? We argue that the answer is what Salas et al, 2005 refer to as teamwork. Understanding how to foster teamwork in software development teams requires an assessment of the team's inner workings as well as the external environment through management buy in (Moe, Dingsoyr and Dyba, 2010).

Studies of critical success factors in agile software development have identified the ability of an agile team to self -organise as one of the critical factors, regardless of the agile practise chosen (Stankovic, Nikolic, Djordjevic and Cao, 2013; Chow and Cao, 2007). The agile manifesto puts the interactions of skilled individuals as playing a more important role than documentation, therefore instead of communicating requirements via documentation, the teams are encouraged to collocate and communicate these requirements face to face. These interactions manifest through the following attributes of agile software development teams:

> ***Colocation:*** colocation facilitates team interactions more efficiently than any other known mechanism. Communication and collaboration among team members is greatly enhanced when team members are within easy reach of each other. Furthermore, the daily, face to face interactions among the team afforded by colocation may result in the transfer or a cross pollination of tacit knowledge (Devos et al., 2008).
>
> ***Adaptability:*** Agile teams strive to accommodate change as easily and as efficiently as possible while remaining cognisant of its consequences. Furthermore, having cross-trained team members increases functional redundancy and, thus, flexibility in dealing with personnel shortages.

Autonomy: Self-managing teams offer potential advantages over traditionally managed teams because they bring decision-making authority to the level of operational problems and uncertainties and thus increase the speed and accuracy of problem solving. This level of decision making is referred to as team autonomy.

Companies have implemented such teams to reduce costs and to improve productivity and quality. Some studies even suggest that this kind of team set up results in higher employee satisfaction, lower turnover, and lower absenteeism. As discussed in preceding paragraphs, the glue that binds all these together is teamwork.

2.5.3. Teamwork

Agile development relies heavily on teamwork. However, there is still a paucity of universal consensus on the definition of teamwork, to this end part of their enquiry into the 'Big Five of Teamwork", Salas et al 2005, have attempted a description based on their extensive analysis of available literature pertaining to teamwork. The description that emerged from the thematic study was that teamwork as '*set of interrelated thoughts, actions, and feelings of each team member that are needed to function as a team and that combine to facilitate coordinated, adaptive performance and task objectives resulting in value added outcomes.* The concept of teamwork will be explored in more detail in the subsequent chapter

2.6. *Theoretical Frameworks*

The importance of sound theoretical roots is imperative to undertaking empirical research in that they help us unearth the truths of software development independent of whatever agile methodology or practise has been chosen. The growth and maturity of Agile as a discipline can only be achieved through an effort to provide robust theoretical framework for the conduct of research on the phenomenon. Due to the paucity of theoretical underpinnings in prior Agile software development research (Dingsoyr et al, 2012), mainly due to the fact that Agile was

borne in practise, agile researchers face a limited sphere of reference when searching for theoretical perspectives. To this end, a few studies have emerged that use various theoretical lenses for the enquiry into certain aspects of agile teams.

In the preceding chapter, literature review, teamwork was discussed as the defining character of the agile software development teams, particularly in Scrum where the seat of action or execution lies in the role of the Scrum development team. It follows therefore, that available theoretical frameworks on this topic be studied. The theoretical framework chosen for this study is the Dickinson and McIntyre's Teamwork Model. This framework is considered most suitable for this study due to the fact that its focus is primarily on the self-organising or self-managing team.

2.6.1. Dickinson and McIntyre Model

The Dickinson and McIntyre's teamwork model, which defines teamwork as those behaviours by the members of a team that engenders communication and a coordination of activities, (Dickinson and McIntyre, 1997) it consists of components such as team orientation, team leadership, monitoring, feedback, back up, coordination and communication. Incidentally, Dickinson and McIntyre (1997) argue that a critical feature of teams is that individuals must coordinate their decisions and activities by sharing information and resources to attain shared goals.

The model consists of three process steps; these are Input, Throughput or process and output. Communication is considered a major component in this model and runs through the entire process.

The first input, considered the second major component in the model, is team orientation which refers to the nature and attitudes that the team members have towards each other. The last of the inputs, team leadership refers to the direction and structure provided by formal leadership and other people.

Monitoring performance, another component considered critical for teamwork, refers to the observation and awareness of activities and performance of other team

members. Feedback is a way for team members to adapt and learn from their performance. Backup behaviour, also considered critical, is about team members helping each other to perform their tasks. Coordination is considered an outcome, a reflection of the team's execution of their activities, the better the collaboration, the better the performance.

Figure 3 below graphically illustrates the Dickinson and McIntyre Teamwork model:

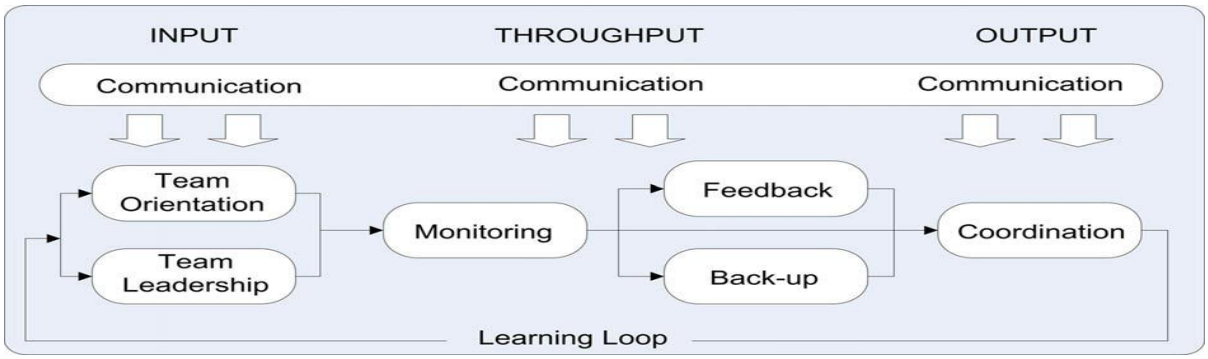


Figure 3: Dickinson and McIntyre Teamwork Models

Dickinson and McIntyre (1997) assert that the higher the levels of coordination, the greater the team productivity. This model has been applied by several studies in an empirical setting on agile teams and has received mixed reviews.

Not all the components of the model were positively linked with improved coordination. One such study was performed by Moe et al (2010). They employed this particular teamwork model in order to test certain Agile practices' impact on team productivity and concluded that the model need to be extended to include trust and shared mental models (Moe et al, 2012, p:489).

The third theory to be considered, Shared mental models theory focuses on the thought process or activities that occur at a team level. The theory proposes that effective team members need to maintain a shared. Understanding of what the goals and expectations are from them as a team, (Yu and Petter, 2014, citing Cannon-Bowers and Salas, 1993.). The shared mental models practices are planning,

reflexivity, leader briefing, team interaction training, self-correction and cross training, (Yu and Petter, 2010).

While their Yu and Petter (2014) study offer good insight into how shared mental models as a theory is applicable to the study of Agile practices, they offer no framework for the said analysis. This, therefore, creates an opportunity for a framework to be proposed and empirically tested as a contribution into the discipline. To this end this study seeks to enhance the teamwork model by Dickinson and McIntyre (1997) through the incorporation of the most relevant aspects of the shared mental models theory as recommended by Moe et al (2010).

The aforementioned study provides a suitable theoretical foundation for the present study to build on and empirically test whether their supposition will bear out in or not. To this end, certain aspects of both the team work model and shared mental models will be merged and a new conceptual framework for the study of Agile Software development teams proposed. The next section will elaborate on the components of the Dickinson and McIntyre and shared mental models and how they come together to form the proposed framework.

Salas, Sims and Burke (2005) have also identified a model that that defines three coordinating mechanisms that are believed to be present in all effective teamwork. These mechanisms are shared mental models, mutual trust, and closed loop communication. Figure 2 below depicts the Salas model.

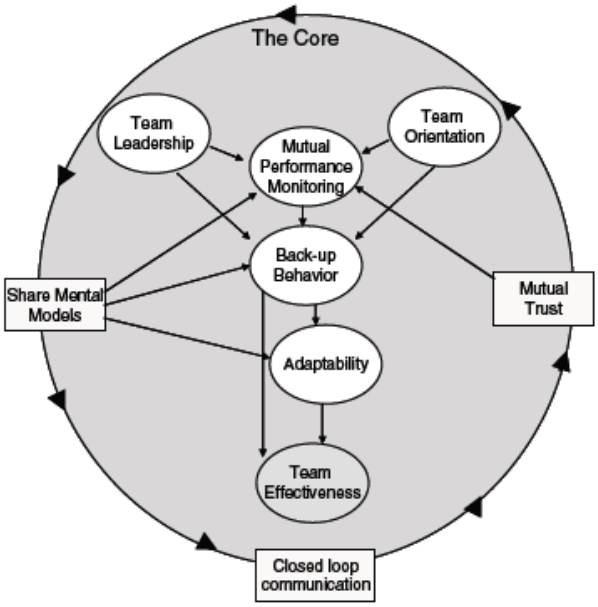


Figure 4: Team effectiveness (Salas et al, 2005)

The Salas et model is a conceptual proposition that has not been empirically tested, but instead offers a proposition for future research. This study seeks to partially heed that call by incorporate some of the mechanisms into this study's research framework together with Dickinson and McIntyre teamwork models.

2.7. *Conceptual Research Framework*

Of the seven components of the Dickinson and McIntyre teamwork models that have been empirically tested, only three have proven beneficial in practice and have been identified by Moe et al (2010) to have an influence on the productivity of self-managing teams. These components are: Orientation, Monitoring and Collaboration. The authors have also identified lack of trust and shared mental models as limitations of the teamwork model.

In their study on understanding software development practices using shared mental models, Yu and Petter (2010) posit that trust and reflexivity would contribute to higher levels of team productivity. In line with Moe et al, (2010)'s recommendation for future researchers to extend the teamwork model to include trust and shared mental

models, and also in an effort narrow down the scope of inquiry into influencers of enhanced productivity by self-organising, agile development teams, a conceptual framework is therefore proposed that extents the teamwork model with trust and reflexivity (components of shared mental models).

We deduce from Dingsoyr and Dyaba's (2012) study on team performance that the combination of both the teamwork model and shared mental models provides the best lens through which to explain how self-organising teams produce enhanced levels of productivity. To this end, we shall endeavour to conceptualise an alternative teamwork model as depicted in figure 2.

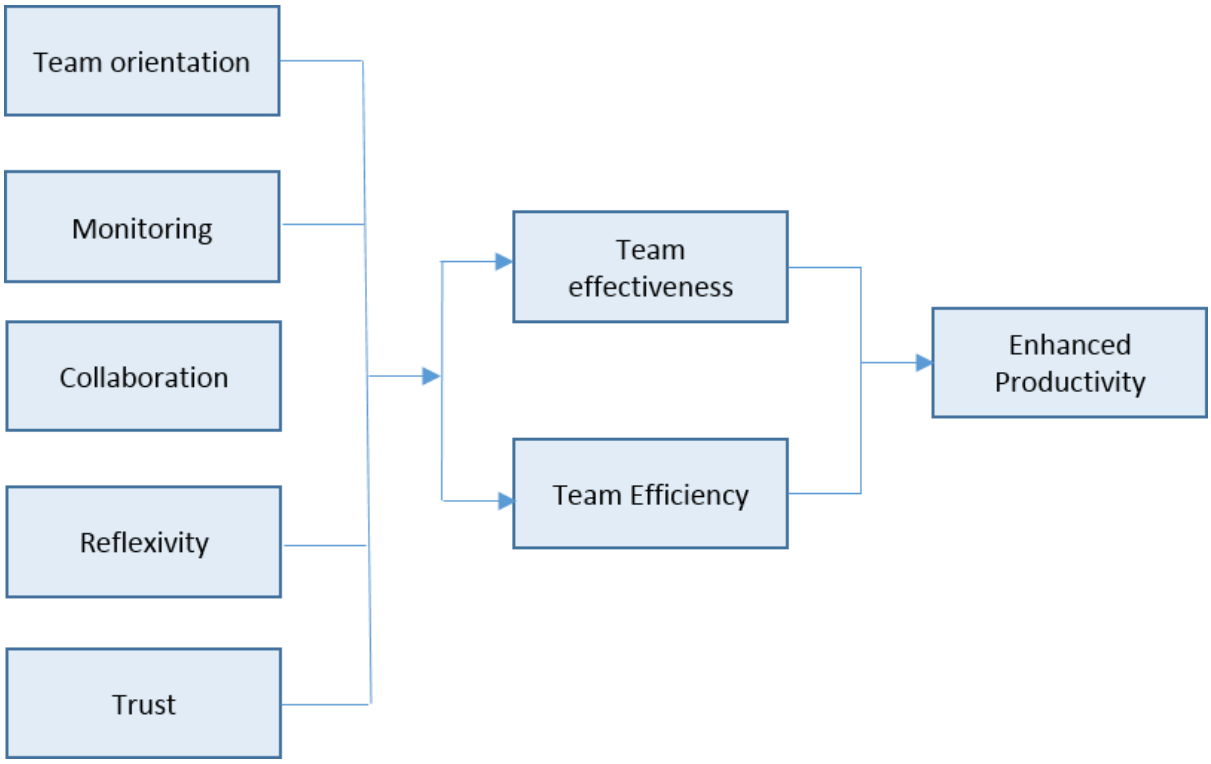


Figure 5. Conceptual research framework

The elements of the conceptual research framework are discussed next.

Team orientation: We have already established in preceding paragraphs what team orientation is as well as the fact that it is one of the elements of teamwork measurement that has been empirically verified by Moe et al, 2005. In the context of this study the meaning of team orientation remains unchanged.

Monitoring takes place a different team members take time to review each other's performance. The role that monitoring plays remains unaltered for the purpose of this study.

Collaboration in the context of this study refers to the interpersonal relations that the team members possess. How the individual team members work together to complete their tasks?

Trust in this context refers to the extent to which team members have confidence in their fellow team members to deliver their part, to not let the team down. It also may include the perception of safety in disclosure. Can team members trust each other with sensitive information?

Reflexivity in this context refers to the practise of solicitation of feedback, by team members from team members in terms of improvement gaps. It includes taking time out to reflect on past events and proposing improvement initiatives where appropriate.

Team effectiveness is described by Salas as referring purely to the outcome of the team's tasks regardless of the manner in which those tasks were performed.

Team Effectiveness means delivering the expected outcome.

Team efficiency refers to how the team achieved the outcome. The process that the team went through in order to deliver the output.

2.8. *Summary of Chapter*

In this chapter we have learned that agility in software development was pursued in light of dissatisfaction with traditional methods, largely due to the rigid nature of these methods. The high dissatisfaction with these methods is attributed to the refocus placed on planning, processes and documentation. The adherence to these tasks may be interpreted as taking the focus away from code delivery to compliance with methods and standards. We have also proposed a conceptual framework to provide a lens through which the empirical aspect of the study may be undertaken.

Chapter Three (3): Research Methodology

3.1. *Introduction*

This chapter discusses the research methodology employed for the study into influencers of enhanced productivity in agile software development teams. The different research paradigms, designs and analysis methods will be discussed; appropriate approach chosen and the justification for doing so are offered.

3.2. *Research Paradigm*

A research paradigm is described by Kuhn (1962) as a 'world view' or a set of assumptions made regarding how things work. Information systems research has at its disposal, multiple paradigms that may be adopted in order to conduct research. Three such paradigms have been identified as by Orlikowski and Baroudi (1991) as interpretive, positivist and critical paradigms or philosophies. A brief description of each philosophy will be offered in the section, together with an explanation of why it was or was not applied in this study

3.2.1. Positivism

Ontologically, positivism assumes that an objective physical and social world exists independently of humans and that this physical and social world can be fully understood. Positivism further assumes that human action is intentional and rational, that their interactions are stable, orderly and relatively conflict free. Any detection of conflict is interpreted as an anomaly that should be corrected in order to maintain the status quo. In positivist research, context does not necessarily play a role in the research (Orlikoswski and Baroudi, 1991).

Epistemologically, positivist research is deductive in nature and is concerned with empirical falsification or verification of theory about a particular phenomenon. This deductive nature of positivism also assumes that the researcher to be impartial in the assessment of the phenomenon of interest. Understanding this phenomenon is gained through the pursuit of measurement, the construction of instruments that capture the essence of the phenomenon. Based on this paradigm, the role of the

positivist researcher is to determine the objective physical reality by devising instruments that are able to identify and measure those dimensions of reality that are of interest to the researcher (Orlikowski and Baroudi, 1991).

Justification for not applying positivist paradigm:

Due to the fact that the purpose of our study is to understand *why* and *how* agile software development team's self-organizing principle enables enhanced productivity, as opposed to measuring the prevalence or the significance of this phenomenon or determining the direction of the causal relationships between agility and productivity, a positivist approach is therefore unsuited for this study. Positivism in this particular case study will only not disclose what has not already been established, that there is a positive link between the adoption of agile software development and enhanced productivity (Dybar and Dinsoyr, 2008); (Moe et al, 2001). The current study sought to explain how and why this phenomenon is occurring.

3.2.2. Critical Philosophy

As a paradigm, Critical philosophy's ontology is that everything is rooted in history and therefore nothing exists in isolation. It also assumes that social reality is produced and reproduced by humans but that there are objective properties that tend to dominate human experience (Orlikowski and Baroudi, 1991). This paradigm assumes that society is not limited to existing in confinement and thus advocates for unfulfilled potentialities.

Epistemologically, it attempts to uncover restrictive and alienating conditions of the status quo and seeks to emancipate (Bhattacherjee, 2012). Critical philosophy assumes that the role of the researcher is that of bringing to consciousness the restrictive conditions of the status quo (Orlikowski and Baroudi, 1991).

Justification for not applying Critical Philosophy

It is for the above epistemology that this approach is deemed unsuitable for this study, this paper seeks to uncover the reasons behind the status quo, not to alter it.

3.2.3. Interpretivist Paradigm

A third philosophy, Interpretivism on the other hand is the belief that the social world is not predetermined, but rather produced by and sustained by humans through their actions and interactions. This paradigm assumes that the social reality cannot be determined or discovered, but rather interpreted through observation of social interactions. This paradigm further assumes an endowment of subjective meaning to social interactions. These meanings are however not expected to remain static, as they are formed, transferred and used, these meanings are also negotiated, which implies a certain level of fluidity in the interpretations as reality shifts and circumstances change (Orlikowski and Baroudi, 1991).

Epistemologically, Interpretivism asserts that understanding social reality requires that one understands how practices and meanings are formed by the language and unspoken rules shared by humans working towards some common goal (Orlikowski and Baroudi, 1991). This paradigm posits that in order to understand a phenomenon of interest, one need to get inside the world of those generating it. The interpretivist paradigm makes no distinction between the social practises undertaken by humans and the languages used to describe those practises.

Justification for adopting interpretivist paradigm

Because this study aimed to understand ‘how’ and ‘why self –organising teams are perceived to be more productive, we believe that an interpretive approach is most suitable in achieving this end. From this perspective we wish to gain insight and understanding, and find reasons for the phenomenon, within its own context. This is consistent with the assumption that knowledge of reality is gained only through social constructs such as language, consciousness, shared meanings, documents, tools and other artefacts (Klein and Myers,1999) cited Kaplan and Maxwell (1994). It is these social constructs that are believed to have an ability to explain how Agile Software development teams are able to achieve enhanced levels of performance.

3.3. *Research Design and Strategy*

3.3.1. Research Design

Because interpretive research engages subjects involved in a phenomenon in order to find subjective interpretations of social phenomenon (Bhatacherjee, 2001), the study will assume an explanatory research design which will enable us to explain how the agile software development teams are able to produce enhanced levels of productivity.

3.3.2. Research Strategy

Bhatacherjee (2001) identifies two types of interpretivist designs that would be suitable for this study; that is, case research and ethnography. Case study research is supposed as possessing a strong ability to discover a wider variety of social, cultural and political factors related to the phenomenon of interest not known in advance.

Ethnography on the other hand emphasises the importance of the cultural context of the phenomenon, which requires that the researcher be immersed in the culture over an extended time period. While suitable for this type of enquiry, ethnography is not deemed to be a practical option due to time and cost constraints. Therefore, case study research was the strategy employed in the study of agile software development team phenomenon.

3.4. *Case Study*

According to Yin (1991), a case study is an ‘empirical enquiry that investigates a contemporary phenomenon within its real life context, especially when boundaries between the phenomenon and context are not clearly evident. He continues to assert that case studies are most suited to answering the how and why type questions, where the control of behavioural events is not required and that the focus is on a contemporary event.

Through the literature review we have established that Agile software development has consistently delivered desired results, we have also learned that the self-organising teams have played a critical role in the success of these practices, however we have not yet established how these positive results are achieved. Therefore this study aimed to uncover the factors that influence the performance and productivity levels of self-organising teams; thus a descriptive case research was best suited for this study since this is the type of case study that seeks to find explanations and description for an observed phenomenon (Bhattacherjee, 2012).

Consistent with the objectives of the study, which seeks to understand and explain the above mentioned phenomenon in its own natural setting, a single site case study is deemed adequate to this end, this is due to the fact that this study does not seek to generalise, for in that case multiple case designs would be appropriate.

The *unit of analysis* in this research is the agile software development ***team***. In order to gain a broad as well as deep insight into the influence behind the productivity levels of this team, both individual team members and focused group type interviews will be conducted.

The team that has been identified, as described in the preceding sections consists of the following roles: the customer (product/system owner), business analysts, process analysts, IT architects, developers, testers and senior users. Only a single team consisting of 18 team members was identified to partake in the study. This sample size is consistent with what is considered sufficient in studies whose aim is to gain an understanding of the commonalities within a fairly homogenous group.

3.5. *Case Description (Study location and context)*

The case study was carried out within the context of one of the major banks in South Africa. Due to a request for anonymity, the bank is henceforth referred to as *ABC Bank.* ABC bank is a large financial institution with a Retail division that offers a range of banking and other financial products to both personal and business

customers. The bank has a large foot print across the country with the number of branches exceeding 700 in number. The retail division alone has acquired more that 5 million customers. The bank's current core banking platform is old and outdated and therefore the bank has made a decision to replace their ailing technology with a modern technology platform called SAP several years ago.

The project has been stalling and the organisation was getting impatient. ABC bank has made the switch to Agile software development as a means to fast track the delivery of a new core platform that will enable the bank to compete in a fast paced banking industry.

As mentioned in an earlier paragraph, the project has been running for a while and the bank saw it fit to replace the old leadership with a new leadership in line with the change of delivery approach.

Due to its extensively documented success rate, the Agile practice adopted by ABC Bank is the SCRUM Methodology. It is also adopted because of its focus on project management, a discipline which the bank finds to be of grave importance.

At the beginning of the financial year, the following goals were set out for the team, by the bank:

- Scope: The team was expected to deliver 11 Savings and Investment products on to the SAP Platform. These products were delivered with full functionality, as defined by the product owners. Success was measured by the number of defect free products in the customer's hands (project charter). While this is a complex set of products, there is a significant amount of overlap in products features and some functions that will allow some of the software components to be re-used.
- Timelines: the products were delivered in 7 sprints within 12 months. The releases were structured in 4-8 week cycles depending on complexity. A sprint would constitute a minimum of one full product with 80 percent of the functionality released to pilot sites to test with customers. Any issues

experienced or any defects raised would be fixed, tested and once successful, the functionality would be rolled out to all the branches nationwide.

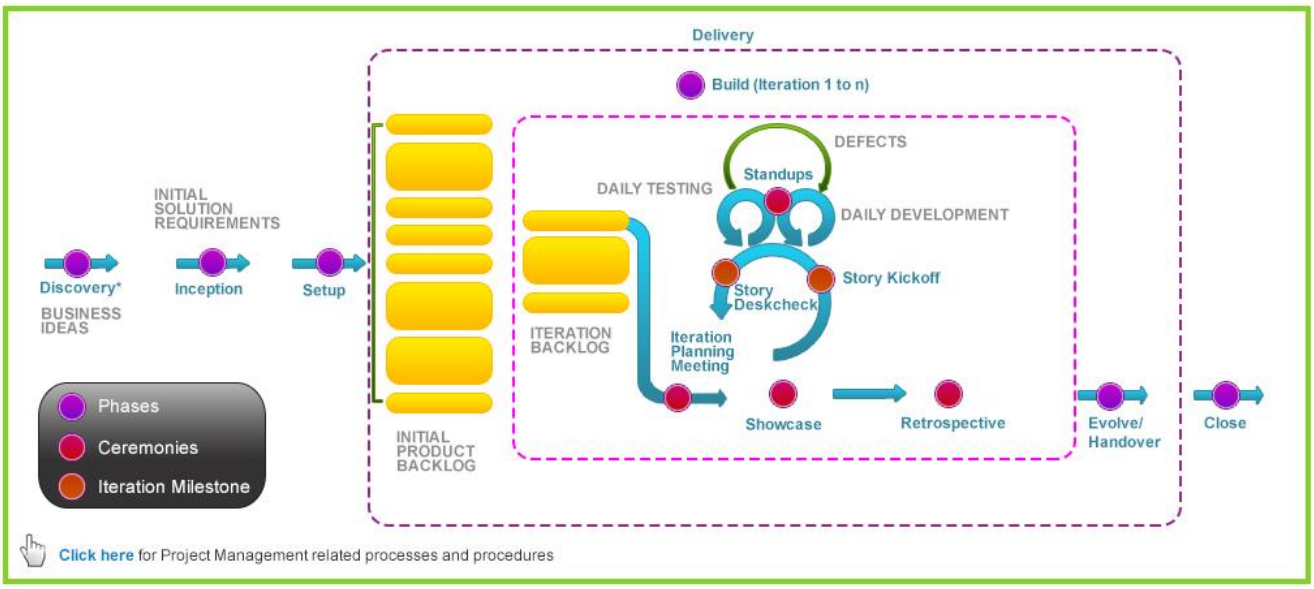
- The budget allocated was R105 million. The budget was centrally held and allocated per quarter depending on progress.

Team Structure:

The agile software development team was structured within the context of Scrum, consisting of both business and IT representatives. The agile software development teams consisted of the following roles:

- Product owner: prioritises the backlog
- Business analysts: defines the functions and features of a product for the development team
- Testers: provides assurance that the functionality is working as designed
- Scrum master: fulfils a project management role
- Developers: build the software functionality according to defined development standards.

The core banking software development team members were ***collocated*** in the same conference room that was dedicate to them for the duration of the project. They made use of whiteboards and walls to write up all there was to know and do about the sprints. Every available surface (whiteboards, walls even glass windows) were used to scribble important information, discussions and talking points. The following process was followed by the team:



1. **Discovery:** the purpose of this step is to validate an idea or understand the customer's problem. During this step, the business need is discussed and captured, research on external customers is carried out and existing data is analysed. The requirements then get consolidated and prioritised. The output from this step is the production of the business case and/ the project gets authorised and proceed to the next stage.
2. **Set up**: the purpose of this step is to set up the team and environments in order to start delivery. the agile software delivery consists of both core team members and non-core team members. The core team members are the people that are dedicated 100% to the project whereas non-core team members are people whose contribution to the project does not require 100% dedication, can be allocated to multiple projects and maybe made available as and when needed. Once the team has been on-boarded, then tools and standards are discussed and agreed. The logistics are organised (co-location, whiteboards, hardware licences etc). Agree on reports and ceremonies.
    a. Reports may include burn up charts, daily status reports, feature completeness
    b. Ceremonies include Iteration planning, Sprint planning, Daily stand-ups, Sprint review/ Showcase, Sprint retrospectives.

3. **Inception:** the purpose of this stage is to define scope of the project/ initiative. To create a shared vision of the project. Also a technical analysis of existing bank systems is undertaken. The technical approach is agreed (build or buy off the shelf), what the test strategy will be, the cost of the project based on the current understanding of the scope, governance model, acceptance criteria etc.
4. **Develop and Deploy**: this is an iterative stage where the project is broken down into small manageable, time boxed sprints. The delivery is aimed at building and demonstrating small, valuable pieces of work that are ready for release in each sprint. The purpose of this is to get feedback from the user as quickly and as often as possible to e ensure that the products continues to meet user expectation. This feedback is solicited through the showcase ceremony where the user gets to experience, touch and feel the product/ feature and provide their feedback.
5. **Evolve/ Handover**: The last step in this process involves the transfer of ownership to the users to form part of their business as usual. This step is often carried once the product/ feature has been successfully deployed and achieved stability for a defined period of time. All relevant and agreed documentation is produced together with the project metrics and any preliminary benefit tracking reports.

The team would commence each ***sprint*** by defining a backlog, grooming it and deciding on the final scope of the sprint. The sprints were groomed around and named after anchor functionality, which may be either a complete product like a Fixed Deposit Investment or major functionality like Account analysis function. The size and complexity of the sprint would determine the length of the sprint. Although it may be common practise that a sprint has to last 4 weeks or be groomed further, it wasn't always possible with this team, firstly due to the fact that success is measured by products/ functionality in customer's hands, therefore, some of the sprints had to be lengthened in order to afford sufficient time for the team to deliver value to the customer. However, the team has scheduled show cases on the first Monday of each month in order to demonstrate any progress made thus far to the leadership of the organisation.

The team would meet daily for 15 minutes for a stand up to discuss three things, the outcome of the previous day's activities, the activities to be carried out today as well as any issues that needs to be ironed.

Immediately after the release of each sprint, the team would have a retrospective session where lessons learnt were discussed. The team would then undertake to continue doing what has gone well and stop doing what has not gone so well. Every team member is expected to participate and contribute to these retrospective sessions.

3.6. ***Data Collection***

Due to the interpretive nature of the study, qualitative data as collected, and a non-probability *sampling* method employed for the selection of case study participants. Saunders, Lewis and Thornhill (2009) discuss a range of non-probability sampling techniques that may be selected based on the nature of the study being undertaken. One such technique is termed purposive or non-judgemental sampling with a focus on heterogeneous sampling.

This sampling technique was adopted for this study because it is not necessary for the sample size of this research to be representative of the agile software development population. Instead it is meant to only identify key themes that will help explain how agile software development teams achieve enhanced productivity. The technique further enables the researcher to exercise their own discretion when selecting participants in a way that best answers their research question and meet the research objectives.

Yin (1991) has identified at least six sources of data that can be collected for a case study. These sources include; documents which consists of written material like minutes of meetings, emails and memos; archival records that include organisational charts and financial records; open ended, structured or focused, face to face interview and direct observation (Benbasat et al, 1987).

Interviews have been identified as the most important source of case study information by Yin (1984). Interviews are characterised as personalised form of data collection method that are conducted using an interview protocol (Bhattacherjee, 2012). Consistent with this reason, semi structured interviews were the data collection methods employed for this study.

A semi-structured interview method has been applied and the interview questions answered by participants were based on a list of themes extracted from the study's research questions and conceptual research framework. While the researcher will has stuck to the line of enquiry, more insight could be gained when the respondents were encouraged to offer opinions over and above just answering interview questions, as has been noted by Yin, 1984.

The interview guide consisted of five objectives, in line with the objectives of the study as stated in the first chapter; that is, each objective had several open ended interview questions related to it. The interview participants were requested to answer each of the questions in as much detail as possible. There were 13 questions in total and each participant answered most of the questions. The interviews were conducted at the bank's premises in a private meeting room away from the project room. 18 respondents participated in the individual interviews. Not every participant was comfortable with the session being recorded.

The average duration of the interview was about 30 min depending on the verbosity of the respondents. The researcher paid close attention not only to the words that were said in response to the interview question but also the tone of voice as well as non-verbal language and cues by the respondents. This is deemed important for interpretive case studies as it captures the essence or the subjective meaning of the phenomenon within its own context.

The researcher is aware of and appreciates that there are some risks inherent in using interviews, these risks include response bias, inaccuracies due to poor recall

and also respondents telling the interviewer what they believe the interviewer wants to hear (Yin, 1984).

To mitigate the identified risks, and for the purposes of triangulation, content analysis has also been performed in order to further uncover any information that converges with that elicited through interviews. Typical project artefacts were sought and subsequently received, these included, minutes (feedback) of meetings, memos, announcements, scope documents, risks and decision logs, change requests, decision logs, weekly status reports by the scrum master, etc.

The data collection took place towards the end of the year when most of the work required of the team was completed, however the events of the past year was still foremost in the minds of the team. This was observed by the researcher during the tour of the project room that was given to the researcher. The walls of the room were still full of colourful notes, scribbles and stickies of all shapes and sizes and a visibly positive recollection of what the artefacts on the walls represented was demonstrated.

Regardless of the sources of data, there are certain principles that need to be observed into the enquiry in order to improve the quality of the study, (Yin 2001). These principles are identified as follows:

- Using multiple sources of evidence to *triangulate* the data; this is the use of more than one source of data aimed at corroborating the same fact or phenomenon. This approach strengthens the construct validity of the study.
- The *second principle* is the use of a case study database, a repository that contains the study's raw data that may be in the form of notes, documents, tabular materials like questionnaires and narratives, the benefit of doing this Yin (1984) asserts, is an increase in the reliability in the case study.
- The *third principle* that need to be observed in order to further increase the reliability of the conclusion and any inferences drawn by the researcher is that of maintaining the chain of evidence. This is the

traceability of all the steps taken to conduct the case study from the research question itself through to the report. Any reader should be able to trace the evidence back and forth through this chain of evidence (Yin, 1991). The report will also be presented back to the members in an effort to further establish the validity of the findings.

3.7. *Ethics Consideration*

This study was undertaken with strict observation of the University's code of ethics and the bank's code of conduct. The permission of the head of the department in question was sought prior to any participants being approached. Participation in the study was only on a voluntary basis based on informed consent. The identity of the participants is kept strictly confidential and may be guaranteed in writing.

Data collected through interviews is kept confidential and protected from unauthorised access. The participants are assured of the study being only for research purposes and shall not in any way be used to disadvantage the participants. Therefore, should the organisation request the report for whatever reason, this would only be done with the participants' prior consent.

3.8. *Limitations of the study*

The main limitation of this research is the use of a single case study design. The single case is and of itself sufficient for the objective of this study, and does not include generalisation, however it obliterates any possibility of replication of the results.

The second limitation is the possible respondent bias due to the propensity of the respondents to tell the interviewer what they believe the interview wants to hear, which may affect the accuracy and the reliability of the analysis.

3.9. *Summary of Chapter 3*

In this chapter on research methodology, interpretivist paradigm was discussed as one suitable for this study. The chapter also discussed the choice of case study as a research strategy and the reasons for it. The description of the case was offered. The chapter also argued that the most suitable data collection method should be interviews, both structured and semi-structured. And finally the chapter discussed ethics considerations as well as limitations of this study. In the next chapter, data collected is analyse and the findings discussed.

Chapter Four (4): Data Analysis and Discussion of Findings

4.1 Introduction

Different analysis method can be applied to any study depending on the objectives as well as the context of that study. Several qualitative data analysis tools or methods have been considered for this study. Bhatachejee (2001) identifies these methods as Grounded theory: a technique development by Glasier and Strauss (1967) as a process of classifying and categorising data segments into a set of codes, categories and relationships and interpretations are solely based on observed empirical data. This approach makes no provision for pre-determined or pre-existing theories and it is for this reason that grounded theory is not the best option for this study due a pre-existing theoretical framework associated with this study.

4.2 Thematic Analysis

Thematic analysis is qualitative analytical method used to identify analyse, organise and categorise qualitative data into themes or patterns of a phenomenon. Thematic analysis is a fairly flexible method that may be applied both inductively and deductively based on the author's theoretical position (Braun and Clarke, 2016). The inductive perspective means that identified themes are directly linked to the data, and as such precludes any pre-existing theory. A deductive approach on the other hand, deductive or theoretical thematic analysis is driven by the study's theoretic interest in the area, the latter analysis perspective is one to be adopted by this study due to the fact that this study does possess pre-existing theoretical disposition. It is important that the author emphasises that the deductive approach in this instance is distinct from the deductive nature of the positivist research paradigm.

Available literature on thematic analysis does not offer any consensus regarding how this process should be carried out, (Braun & Clarke 2006; Aronson 1994; James and Harden, 2007). This study adopts the steps outlined by Braun and Clarke due to its

comprehensive nature. Most of the studies outline only 3 steps in this analysis process while the said authors outline six.

The analysis of the research data was carried out as follows:

Step 1*: Data familiarisation*: in this step the researcher has manually studied the data collected through semi structured individual interview and documentation a few times as expected. The outcome of the data familiarisation stage was list of ideas that piqued the interest of the researcher. The list of ideas seemed haphazard and incoherent at first. But with repeated attempts, the quality of the ideas generated improved. Most of the interesting ideas emerged in subsequent readings. The more the researcher reviewed the material, the more crystallised the ideas became.

Step 2*: Coding:* the second step in this process entails the identification of preliminary codes from the data. This could be features or elements of the raw data that can be analysed most meaningfully. The outcome of the coding stage was a long list of codes that may be arranged into themes

Step 3*: Searching for Themes*: the third stage involved arranging the codes generated in the previous stage into potential themes and organising the relevant coded extracts within the themes that have been identified. The outcome of this stage is a list of themes and subthemes and also residual codes that did not seem to roll up into any particular theme.

Step 4: The fourth stage of this thematic analysis entails *reviewing the themes:* this phase involves two levels of reviewing and refining themes. Firstly, the themes must be reviewed at the level of coded extracts to establish coherence. Secondly, once coherence between the themes and the codes that make it up is established, the themes are then reviewed in the context of the entire data set in order to establish accurate representation of the data set as a whole.

Step 5: The fifth stage of the thematic analysis process entails Defining and Naming themes: this stage entails ongoing refinement and defining the themes that will be presented as part of the analysis. The outcome of this step is a set of fully worked-out themes

Step 6: This step entails the undertaking of the final analysis and writing up of the report based on the outcome of the previous steps.

4.3 Analysis of data collected

In this chapter the research findings, in the context of the research questions, are discussed. The chapter consists of 4 subsections where each of the research questions will be addressed individually as per the outcome of the empirical data.

4.4 Participant demographic

Below are the roles fulfilled by the research participants:

Product Owner (PO) X1
Scrum Master (SM) X1
Business analysts (BA) X3
Quality Engineers (Testers) X6
Software Engineers (SE) X6
Process Analysts (PA) X2

The researcher followed the thematic process outlined in the previous chapter. The interview questions that the researcher asked were informed by the conceptual research framework and the study's theoretical underpinnings.

4.5 Analysis of qualitative data collected

This chapter set out to explain the attributes that are responsible for the success of agile software development teams in relation to the literature review as well as theoretical underpinnings. Each objective/theme will be discussed as follows:

- Theme A seeks to describe agile team formation - Research Objective 1: To describe how Agile Software development teams are set up.

- Theme B seeks to describe team capabilities - Objective 2: To determine the capabilities that characterises the team.

- Theme C seeks to describe productivity manifestation - Objective 3: To determine how these attributes manifest into enhanced productivity.

- Theme D seeks to describe the impact of external factors on productivity - Objective 4: determine any external factors that may contribute to enhanced productivity

Theme A: Research Objective 1: To describe how Agile Software development teams are set-up:

Interview Question: Describe how the Agile Software development team was formed/set up?

PO: '*Agile software development teams are set up according to a positive match between the products to be delivered and the skills the individual team members posses'*
SM: '*The team is formed based on the scope of work, skills fit as well as personality fit with the rest of the team. It is important that team members be compatible'*

BA: '*The team members are chosen based on their understanding of the functions that supports the objectives to be achieved. The team must consist of both subject matters experts in order to ensure quality and non-subject matter experts in order to facilitate knowledge transfer and continuity'*

SE '*The team was formed around the skills and competencies required to deliver the solution'*

QE: '*The team was formed by the people who possesses the right knowledge and experience required to deliver the solution'*

PA: '*the team is formed based on the fit between the requirements and skills '*

Researcher's Comment

The formation of Agile teams appears to be centred around the possession of relevant, specialised skills required to achieve a specific objective. An agile team was described first and foremost as a cross functional team of skilled people with a common goal. This goal or objective then becomes the team's common purpose, *their raison d' etre*. In the case of ABC bank, the common purpose for this Core Banking software development team was to accelerate the delivery of Savings and Investment products onto the new platform. A variety of skills are required in order to realise the specific goal. The team members are constituted based on the types of skill or skills that each member possesses.

Theme B: Objective 2: To determine the capabilities that characterises the team

Research Question: Describe the skills or competencies required within an agile software development team?

PO: '*as a product owner, I am expected to possess product knowledge, financial processes knowledge, understanding of legal and Tax requirements for each product. The ability to clearly articulate the requirements is directly dependant on the knowledge I have of the product/ problem*'

SM: '*project management/ scrum master skills, development, business analysis, testing are all the capabilities that the team must possesses as a collective'*

BA: '*capacity to understand and appreciate the business rules, product features and processes that enable the solution to deliver value, furthermore for new products, we also request the assistance of the communication and training week'*

SE: '*knowledge and experience in software development practises, tooling and troubleshooting skills'*

QE: '*test automation skills, product knowledge, reporting skills'*
PA: '*process mapping skills, publishing and experience with various process analysis tools'*

Researcher's comment:
Collective technical competency was evident within the team. All the mainstream software development roles were represented as follows: Software engineering and configuration skills, Business analysis skills, Product development skills, People change management skills, Training material development skills, System integration testing skills, quality assurance, Scrum master, software architecture, Process architecture.

Theme C: Objective 3: To describe how capabilities manifest into enhanced performance:

Interview question: Describe the role played by interpersonal skills towards success/ failure of the team objectives?

PO: ' *the ability of the team to work as a unit and good communication skills played a significant role in the success of the team, whenever we identified issues, we always discussed them face to face until everyone was back on the same page'*

SM: '*being able to collaborate as team members played an important role because we were able to collectively move the work forward'*

BA: *'as a team we were faced with many uncertainties, especially at the begging of a sprint, but the team learned to adapt to environmental changes, the more adaptive they became, the more resilient they became'*

SE: '*the interpersonal skills that we found most important were tolerance for each other's viewpoints, humility, face to face communication followed by email in order to keep record, decisiveness. It was very important that whenever a decision is required, it gets made and communicated within a matter of hours, we have learned that waiting for a decision actually holds up the work'*

QE: 'The *ability to effectively manage and resolve conflict was very important. Although it was hard at first, the team got better at it, and as a results, we had less conflict and more robust discussions that were not really taken personally anymore'*

PA: '*the ability to communicate face to face during daily stand ups and collaborate helped to deliver a good solution'*

Researcher's comments:

Communication in general, face to face communication in particular seems to have played a vital role in the success of the team. Other interpersonal skills like conflict management, quick decision making, collaboration, tolerance and adaptability seems to come out in the data. The combination of the above technical as well as interpersonal skills seems to enable the team to self-manage and organise without much need for external leadership. The presence of both business and technical team on the project ensures that there are no external dependencies especially as it pertains to decision making.

Theme D: Objective 4: Determine factors that may contribute to enhanced performance

Research Questions: describe the kind of support that you received from the larger organisation?'

PO: 'the *organisation invested in agile training for individual members of the team, this training helped familiarise us as business representatives with the agile methodology and what is expected of us'*

SM: '*the organisation has significantly reduced the red tape typically associated the project governance, the budgeting process still has some red tape, submissions have to be made on a quarterly basis for funds to be released to the project'*

BA: *'because the work allocated to our team was considered strategically important, we were provided with adequate resources in order to carry out the work, we didn't have to wait too long for resources allocated to us'*

SE: '*the organisation still expects us to do some governance and have not completely removed gated reviews, we still have to log change requests before we can move code into various environments'*

QE: '*the organisation empowered us as a team to deliver the solution as we see fit. The level of autonomy we had allowed us to decide how we do our work, make our own decisions and take full ownership of the success of the team'*

PA: *'we are still expected to model our processes on the Aris tool and submit them to someone else outside the team for review and sig off before we ca publish. This can be a time consuming process'*

Researcher's Comment

External support from the greater organisation in the form of agile training and coaching came out strongly in the data. While the organisation's *funding model* has been adapted to allocate funds at a team level, there is still a lot of command and control associated with the releasing of funds. Another factor that seem to be of importance is the fact that while the organisation has embraced agile, there were still a lot of *processes* that were still being enforced by other business units in the organisation that impedes the team from taking full advantage of the agile methodology.

4.6 Summary of Chapter

In this chapter on data analysis and description of findings, we discussed thematic analysis as the chosen method of analysis. We learned that there is no single, uniform manner in which thematic analysis may be conducted. The approach thus chosen for this study is a six step process outlined by Braun and Clarke, (2006).

The participant demographics were outlined and each participant's role discussed in detail. An analysis of qualitative data was outlined into 4 themes, each theme was discussed in the context of the research objectives and described in detail.

In the next chapter we will discuss the interpretation of findings.

Chapter Five (5): Interpretation of Findings

5.1 Introduction

This chapter serves to interpret the study's findings in terms of addressing each secondary research question and subsequently the primary research question of *how do agile software development teams achieve enhanced levels of performance?* The secondary research questions to be addressed pertains to the transitions that take place during the formation of an agile software development team, the salient capabilities that these teams possess as well as how these capabilities manifest into productivity and lastly how does team behaviour affect or inhibit performance.

5.2 Interpretation of findings

Objective 1: The transition that takes place during the formation of an Agile team

Due to the fact that people are at the centre of agile software development practices, it follows that the most critical challenge usually experienced during the formation of an agile development team will have to do with human relations (Gandomani and Nafchi, 2016). What makes agile team formation even more challenging is that over and above individuals being expected to function together as a team, the very same team is formed for the purpose of being self-organising. In their paper on overcoming barriers to self-organising, Moe, Dyba and Dingsoyr (2016) argue that it is unrealistic to expect a group of individuals (team) that has just formed, to automatically know how to work effectively as a team.

The formation of a new agile team varies from organisation to organisation. In the case of ABC Bank, due to its organisational size, the adoption of the agile approach was very gradual, starting with one business unit with the intention to eventually roll it out to the rest of the organisation. In the case of the team being studied, the prospective team members found their way into the team through various mechanisms. Some were approached by the Scrum master appointed by the

organisation, some, mostly software engineers, volunteered to join the team. Due to the fact that the transformation to agile was not adopted at an enterprise level in a big bang approach, we therefore in this study, see a combination of people with agile experience and those without.

Once the leadership of ABC bank had decided on the program's objectives, a Scrum master was appointed by the project board with a mandate to set up the agile project team to deliver on the objectives. The Scrum master identified the core skills required to deliver the solution and a search for employees who possessed those skills ensued.

Once those resources were identified, they were requested to indicate their willingness to join the team, therefore making participation more or less voluntary. Some employees within the organisation requested to join the team voluntarily. Some of the comments made around the team formation were:

'*Some of us knew each other and worked with each other before, so we were ok with each other's past performance'* ~ Software engineer (developer)

'I heard that that project was very important to the leadership of the organisation and I wanted to participate meaningfully and be noticed, hopefully rewarded for my effort' ~ Business Analyst

'I participated because it was an opportunity for me to take on a new challenge and grow'~ Scrum Master

Once the team was identified, a 3-day discovery session was undertaken where the product owner relayed the goals and objectives of the program, its expected impact on the organisation (in this case a 10% deposit growth) as well as the expectation from the team. During those three days, the product owner, the scrum master and the software engineering team interacted extensively in an effort to understand the problem that the business was facing, the business solution to that problem as well

as the software solution that needs to be built in order to enable the organisation to meet its customer needs.

From the data we can make a link between the initial formation of an agile team and a presence of shared mental models. The fact that the team members alluded to joining the team because of their previous acquaintances with each other prior to the inception of the current team, their familiarity with each other's areas of competence as well as their approval thereof, implies that there is a modicum of trust and mutual respect between those team members.

It is also evident from the empirical data that the team is highly specialised, leaving very little room for generalist skills, except maybe for the Scrum master. This was in an effort to reduce the level of ambiguity when the time to allocate task and keep each other mutually accountable came. This approach further assists in ensuring that only the work that is necessary for the attainment of the goal is carried out, saving time and money.

Subsequent to the discovery phase, the next step in the process was to set up the team and environments in order to start delivery. The logistics were organised (co-location, whiteboards, hardware licences etc), the various ceremonies were agreed upon as well. The next step in this process, according to the agreed Agile approach, was the inception phase, a phase in which actual execution may commence.

It can be argued that, even though we have learned from literature that the initial formation of an agile team can be challenging, this did not entirely bear out in practise, in the case of ABC bank, the scrum master did not report facing any major challenges in assembling the team that would deliver the savings and investment program of work

It is therefore appropriate to posit that the transition that takes place during the formation of an agile team may be positive in nature depending on the level of familiarity of the individual team members with each other.

Objective 2: Attributes of an Agile Software development team

The Scrum guide lists the characteristics of the agile ‘development team’ as self-organising, cross functional and autonomous. The individual members of the team are expected to possess specialised skills and are collectively referred to as the ‘development’ team (Scrum Guide, 2016).

The ABC Bank team is aligned to the guide in that the team is self-organising as far as the allocation of task, execution and monitoring is concerned. This is evident in that subsequent to the discovery and set up phases, the team moved on to the inception phase where actual work began.

The same people that attended the discovery sessions also attended and contributed to the inception phase of the project. This is the stage where requirements were solicited from the business representatives, technical impact analysed and sized. The team also determined the number of sprints that the scope would span across, in this case 7 sprints. The first sprint was expected to commence right away.

One of the outputs of the inception phase was a list of the scope items that the team had undertaken over an unspecified duration of time. This list of requirements was referred to as a backlog; the product owner had the mandate to prioritise the backlog items and continuously adding and refining it.

Another output of the inception phase was task allocation, in the case of ABC Bank, the team members picked tasks that were most aligned to their area of competencies. Meaning developers stuck to developing based on the area of software engineering they were most competent in. The testers executed on test cases while Business analysts and the product owner concentrated on managing requirements.

Some of the comments that were made during the interview were:

'I find it important that I take initiative and get on with my tasks before anyone asks, we all agree at the sprint planning what the allocation of tasks is based on our skills, so it's important that I get on with it is as soon as possible'

'I always want to do things right and on time, so as to not let the team down, it doesn't feel good when the team is disappointed in me'

'when you know that team will ask for feedback at the stand-up, I want to be able to give a positive answer, I need the team to trust that I can keep my word. It's very important to me'

Some of the themes that emerged in terms of the attributes of an agile team are:

Autonomy seems to be another output of the above mentioned combination of technical as well as interpersonal skills. The team seems to enjoy a high level of autonomy as a result. They decide among themselves what approach to take when executing on each sprint. They decide how to groom the backlog how the tasks will be allocated and when they are due. This type of autonomy seems to be concentrated at a team level. Due to the interconnected nature of this work, individual autonomy seems to be confined to a minimum, applicable only to how the task at hand will be executed. Over and above that, team autonomy takes precedence. There is a general sense of ownership of the delivery among the team. And because they own the work and feel empowered, they are happy to go the extra mile in order to deliver on time.

Motivation seems to play a central role in this team's success. Their motivation stems from the level of teamwork and team cohesiveness that is prevalent in the data. The sense of accomplishment that the team acquires with every successful deployment or every obstacle that they overcome seems to fuel them to perform better than they did the last time. Thus the team is motivated to continuously improve/ enhance their past performance.

The team has experienced several set-backs along their journey, like persistent environment instability, defects that took days instead of hours to be resolved and

many disagreements among themselves about technical approach and so on. Intervention from a coach was sought and after a retrospective session, the conflict would be resolved and the team would get back on the same page, and on track again.

From the empirical data we also learn that there is a significant amount of uncertainty that the team has to deal with and therefore it necessitated the need to be ***adaptive***. This is evident in the way the team discusses how not all the information that is required for planning is available when needed. Therefore, in order to move forward, the team would make certain assumptions for the purpose of planning. This however requires that the team be very attuned to their environment and be responsive (adapt) to the changes in those environments.

Some of the comments made around adaptability:

' ...when were informed that our current deployment slot had been cancelled due to enterprise DR, we had to either extend the duration of the spring or deploy earlier, we chose an early deployment slot in order to avoid ripple effect on the subsequent sprints'

Resilience was also another capability of this team. The team had been faced with numerous obstacles from environments being down for extended periods of time where no testing and therefore no progress was made. During the environment crisis, the team regrouped, and identified any stand-alone activities that could be brought forward and executed while they awaited the environment stability to be restored. As soon as the stability of the test environments were restored, the team extended their working hours in order to make up for the lost time.

Objective 3: Describe the role interpersonal skills play on team productivity

Because of the different personalities present within any team, conflict management came out as another important interpersonal skill that the team possessed. They

indicated that there were many disagreements within the team in terms of how the sprints would be delivered, particularly during periods of very high stress like testing stage where a lot of flaws in the build were exposed. There was a tendency towards defensiveness but these soon started tapering off after several sprint retrospectives where it was emphasised that collective responsibility should be taken for any defects and that nothing was supposed to be taken as personal indictment.

According to Highsmith and Cockburn, (2001), interpersonal skills like mutual trust, communication management and conflict management, fast decision making and an ability to deal with ambiguity must be present within the team in order for them to succeed in enhancing productivity. It is not evident whether each member of the team need to possess all these soft skills, only that the team be adequately equipped with these soft skills.

Through the thematic analysis we have observed the emergence of themes like trust and reflexivity among team members. Incidentally the presence of these interpersonal skills supports the conceptual framework as factors affecting the agile team's productivity levels. Other interpersonal factors that emerged from the data as discussed as follows:

A *positive disposition and a willingness* to learn featured very strongly in the data analysis. Because the team is co-located, there is a greater transfer of tacit knowledge between team members which resulted in cross pollination of knowledge. The more individual team members learned about each other's roles, the more fruitful their interactions.

Communication played a very critical role during the delivery cycle. It was of utmost importance that each member of the team be up to speed with every development within the project but also within the organisation at large. Communication was carried out through various forms, formal and informal. The formal communication mechanisms were; emails, sprint planning which took place at the beginning of every sprint The daily stand up was a compulsory 15 min session held at the begging of each workday where the issues of the day were discussed, executed on, or resolved.

The showcase is where the work that has been completed thus far gets to be displayed for users to interact with and provide feedback. The sprint retrospectives where the teams reflected on the previous sprints and identify what behaviours or activities needed to stop, which behaviours or activities the team needed to continue doing. The more informal communication mechanisms were impromptu face to face discussions over coffee, memos and stickies on the walls and social media groups.

It seemed important that the team members *trust* each other during the lifecycle of the project. each team member could be trusted with executing their tasks expeditiously and with high standards of quality. Each team member was always aware of the dependencies on their task and what the impact of turning work late or poor quality work was on other team members and ultimately on the sprint. There has been instances where trust was so broken that some team members were asked to leave the team due to the fact that they couldn't be relied upon anymore.

The combination of self-organisation/ management and autonomy seem to be the active ingredients in the success of this agile software development team. The team sets its own pace, its own control measures and keep each other accountable while they have the freedom to choose the best way to execute on their collective tasks. Because the team feels empowered, they do not need any further incentives to go the extra mile. They have become more and more self-motivated. Their ability to succeed on their own terms serves as reinforcement to keep on doing whatever it took to maintain the pace of success.

While the issue of autonomy featured very strongly in this narrative, the challenge of putting individual autonomy above team autonomy seems to require more monitoring and regulation than seemed present in the analysis, especially among more senior members of the team who tend to assume a superior role.

Objective 4: Describe the external factors that may contribute to enhanced productivity

Since the agile development team can never exist in a vacuum but rather an eco-system, it is therefore imperative that the agile team be understood within the organisational context in which it operates (Cockburn and Highsmith, 2001).

It is apparent in the data that while external support from the greater organisation is in place for the agile team, the organisation itself has not fully transitioned to the Agile approach. How the team chooses to deal with the greater organisation will determine how the team will fare in the future. This is the team's first set up as an agile team.

While there is some level of consultation outside the project team for alignment purposes, the decision makers form part of the core team and as such as self-organised. The team as a collective decides on the scope of the sprint, or the backlog, the team decides the sequence of those backlog items and their delivery dates. No one outside the team determines how the team carries out their work and when they should be done. The team members are accountable to each other; it is every one's job to monitor every other one. While this was misconstrued as policing each other at the beginning of the very first sprint, it because less of an issue in the subsequent sprints as the team members got accustomed to keeping each other accountable. The end of sprint retrospectives also went a long way to quell some of the concerns raised around monitoring. the behaviour that was therefore instilled as a result was tasks were delivered timeously and with better quality.

These processes are imposed mainly by the non-core team members who have not been fully embedded in any particular agile team due to the fact that their services belong to functional lines and are shared across multiple teams. These non-core team members include change management, infrastructure shared services, group legal, group tax. Their involvement with agile teams is consultative in nature and therefore lack the incentive to fully adopt the agile way of thinking. These functional areas have not yet divested themselves of the old, process based approach and is therefore seen as anti-agile.

After each sprint, the team has to prepare documentation for the release of funds for the next sprint. While the team appreciates the rationale behind financial governance, the allocation of funds is still not expedient and sometimes work has to stop until further funding is made available.

The team is of the view that transition to agile is especially challenging for employees who have been practising the waterfall method for many years. Coaching and continuous reinforcement are therefore imperative

5.3 Summary of chapter

In this chapter we have presented our interpretation of the of the empirical data findings in alignment with the research objectives. The objective that addressed the transition to agile, i.e the first objective, we learn that the transition to agile was very gradual due to the size of the organisation and that the participation was mostly voluntary. The second objective addressed the attributes possessed by the agile team, these attributes were consistent with the ones described in the scrum methodology.

The third objective describes the interpersonal skills that are to be found among individual team members, these range from positive disposition, willingness to learn, mutual respect, communication and trust. Through the final objective we provide a description of which and how external factors influence the team's productivity.

The next chapter will discuss conclusion and recommendation.

Chapter Six (6): Conclusion and Recommendation

6.1 Introduction

In this chapter we will discuss the implications of this study, the future recommendation as well as the limitations. We will focus on how choosing a specific approach aids the transition to agile, we also discuss the alignment of conceptual framework to the empirical data, what future research should focus on in order to develop this concept further, as well as the limitations that impedes us from making the study more generalizable.

6.2 Implications of the study

With the purpose of this study being to describe how agile software development teams achieve enhanced level of productivity by empirically answering for research questions. It is evident from this empirical study that agile teams do display enhanced levels of productivity, and that there is a positive link between enhanced productivity and self-organizing teams that display the attributes described in the preceding chapter.

The case study in this context was based on a SCRUM agile team with roles ranging from product owner, feature analysts, scrum master and the team. The Scrum guide provides a description of each role in detail as well as the corresponding responsibilities, and we have found the team in this here study to be conforming to the scrum guide.

It is evident in this study that if a team chooses an agile practice, in this case Scrum, and pays particular attention to team orientation, team interaction (interpersonal skills) as well as the external environment, productivity may be enhanced.

From the empirical evidence, it is now understood that the early stages of an agile team formation are critical to the rest of the project journey. Based on the evaluation of the empirical data we can infer that the transition of the team at ABC was deemed successful partially due to the fact that the individual team members had a significant

degree of familiarity with each other based on prior assignments, therefore creating a positive link between team orientation and enhanced productivity.

In the attempt to describe the capabilities that an agile team possess, the findings revealed that specialisation in a particular field is required. This particular element was not presented as part of the conceptual framework, however its contribution to the success of the agile is perceived as important.
The impact of the external environment of the agile team productivity was more neutral in nature. The team looked to the organisation for support, training and funding, which were provided, however the same organisation also imposed some challenges to the team through scheduling conflicts and code contention.

This research report has shown that there is multitude of influencers to an enhanced performance in agile software teams. These factors need to be taken into consideration prior to establishing and agile team for the delivery of a specific project.

6.3 Recommendation of the study

Due to the fact that agile software development was borne in practice, it is safe to say that empirical research into this phenomenon is still lagging behind practise. The advancement of the agile methodology is growing at a rapid pace. New and different variations of agile like SAFe and KANBAN are but a few examples. It is important to point out that this acceleration in the evolution of the phenomena is taking place in the absence of sound theoretical underpinnings and corresponding empirical analysis. This the researcher views as a threat to the discipline.

During the course of this study, we have uncovered many gaps around the subject of agile software development teams. For instance, we understand from the literature and by happenstance within the empirical evidence that self-organisation and team autonomy are mutually inclusive as influencers of productivity levels in agile software development teams, therefore it recommended that this be further investigated. Also out of scope of this study but

requires further investigation is the influence of training the individual members of the team on the specific agile practise chosen by the organisation.

6.4 Conclusion

In conclusion, there is evidence of a positive link between enhanced productivity and agile software development teams. The contribution of this research report to the field of Agile software development research is twofold:

The primary contribution is the theoretical extension of the Dickinson and McIntyre's team work model as recommended by Yu and Petter (2010). The literature review for this study exposes a paucity of sound theoretical underpinnings in agile software development research, of which this study aimed at reducing.

In a few cases where the presence of theory is significant, the studies only stop at conceptualisation. Therefore, over and above the theoretical contribution, this study empirically tested the extended framework, therefore either reinforcing or falsifying the theoretical proposition made by Yu and Petter (2014).

6.5 Limitations of the study and future research.

The primary limitation of this study is the single site nature of the case study and therefore not generalizable. The conclusions drawn by this study are only applicable to the said case study and therefore the study will net to be replicated across multiple sites to make it more generalizable. Furthermore, the case study only focused on a single agile practice, i.e Scrum and therefore may not be generalizable. Due to time and resource constraints, a predefined theoretical framework was conceptualised and therefore very narrow in focus, a grounded theory study on this phenomenon would surface more generalizable, unexpected themes that may display a more conclusive outcome on the link between enhanced performance and agile software development teams.

References

Agile Alliance, 2001. *Agile Alliance.Org.* [Online] Available at: http://www.agilealliance.org/the-alliance/the-agile-manifesto/ [Accessed 07 November 2014].

Alhojailan, M. I., 2012. Thematic Analysis: A critical review of its process and evaluation. *West East Journal of Social Sciences,* 1(1), pp. 39-47.

Benbasat, I., Goldstein, D. & Mead, M., 1987. Case Research Strategy in Studies of Information Systems. *MIS Quaterly.*

Bhattacherjee, A., 2012. *Chapter 5.* Second ed. Tampa, Florida: Creative Commons Attribution.

Braun, V. & Clarke, V., 2006. Using Thematic Analysis in psychology. *Qualitative Research in Psychology,* 3(2), pp. 77-101.

Chow, T. & Cao, D., 2008. A surver study of critical success factors in agile software projects. *The Journal of Systems and Software,* Volume 81, pp. 961-971.

Dingsoyr, T., Nerur, S., Balijepally, V. & Moe, N., 2012. A decade of Agile methodologies: Towards explaining Agile software development. *The Journal of Systems and Software,* Volume 85, pp. 1213-1221.

Guzzo, R. A. & Dickson, M. W., 1996. Teams in organisations: Recent research on perfomance abd effectiveness. *Annu. Review. Psychol,* Volume 47, pp. 307-338.

Highsmith, J., 2002. The great methodologies debate Part 2. *Cutter IT Journal,* Volume 5, pp. 7-18.

Klein, H. K. & Myers, M., 1999. A Set of Principles for conducting and evaluating interpretive field studies in Information systems. *MIS Quaterly,* 23(1), pp. 67-94.

Laanti, M., Simila, J. & Abrahamsson, P., 2013. Definition s of Agile Software development and Agility. *Euro SPI, CCIS,* Volume 1, pp. 247-258.

Moe, N. B., Dingsoyr, T. & Dyba, T., 2010. A team work model for understanding an agile team: A case study of a scrum project. *Information and Software Technology,* Volume 52, pp. 480-491.

Mohammed, S., Ferzandi, L. & Hamilton, K., 2010. Metaphor no more, a 15 year review of team mental model construct. *Journal of Management,* Volume 1, pp. 1-34.

Orlikowski, W. J. & and Baroudi, J., 1991. Studying Information Technology in organisations: Research approaches and assumptions. *Information Systems Research,* 2(1), pp. 1-28.

Saunders, M., Lewis, P. & Thornhill, A., 2009. *Research methods for business students.* 5 ed. Essesx: Pitman Publishing.

Schuh, P., 2004. *Intergrating Agile developmemnt in the real world.* s.l.:Charles River Media Inc..

Vijayasarathy, L. R. & and Turk, D., 2008. Agile Software Development: A survey of early adopters. *Journal of Information technology Management,* 15(2), pp. 1-8.

Yin, R. K., 1991. Case Study Research, design and methods. *Applied Social Research Methods,* Volume 5, pp. 1-5

Yu, X. & Petter, S., 2014. Understanding software development practises using shared mental models theory. *Information and Software Technology,* Volume 56, pp. 911-921.

a. *Appendix A.*

Appendix B: Interview guide questions:

Objective	Interview Question
Determine how Agile teams are set up	• How did you become a member of the project team? • What role do you play on the team? • Which skills do you bring to the team?
Identify attributes/ characteristics of agile team	• Describe your day to day tasks on the project? • Describe your relationship with your team mates • Describe the kind of impact, if any this team had on your current work
Determine how the these attributes manifest into enhanced productivity	• Describe how you approach your work as a team • What challenges have you encountered as a team? • How were these challenges overcome
Identify external factors that may contribute to productivity	• Which stakeholders do you have to engage within the greater organisation • What kind of impact have these stakeholders had on the team's output?