

**DEVELOPMENT OF A FRAMEWORK TO UNDERSTAND THE FACTORS THAT INFLUENCE
SOFTWARE PRODUCTIVITY IN AGILE TEAMS**

by

VIOLA NZOU

submitted in accordance with the requirements for
the degree of

MASTER OF SCIENCE

In the subject

COMPUTING

at the

University of South Africa

Supervisor: Professor E. Mnkandla

Date (October 2017) submitted

ABSTRACT

Productivity improvement in the software industry is one of the major challenges facing many software development companies in this century. Most companies have adopted agile methodologies in order to profit from the benefits claimed for them. Agile methodologies are characterised by frequent software delivery, short feedback loops, quicker response to change, and problem identification earlier in the development process.

The agile approach has been recognised as paving a way for companies to acquire higher software productivity, delivering good-quality and cost-effective software, enabling software development companies to respond to business challenges with their demands for high quality, high performance and high development speed in delivering the final product. For companies that adopt agile methodologies, understanding the factors that influence their teams' software development productivity is a challenging task for management and practitioners today.

In this research, an analysis is presented that identifies productivity factors that affect agile teams. It is a study of agile methods to identify common agile practices and/or values that have impact on productivity, and describes suitable metrics that could be used to measure agile team productivity. A qualitative research approach was used, and the case study was chosen as the research strategy. Two South African companies that are located in two different provinces and that adopted agile methodologies in their software development, were selected for the case studies. Qualitative content analysis was used in the research to permit subjective interpretation of factors that influence agile team productivity, and to analyse to what extent these factors affected productivity.

This research has shown that an understanding of the factors that influence an agile team's productivity gives significant insight into the way agile teams work, motivates team members to work together, and leads to uniform metrics in tracking each team's progress. The study indicates that tracking an agile team's work and providing adequate tools needed to execute their tasks results in improving agile team productivity. It should be recognised that using metrics to measure performance in agile teams is helpful in creating a team's culture and trust.

In this study, it was found that the factors identified in both literature and case studies affected productivity in the two companies under study, both positively and negatively. The study also found that applying the correct metrics in assessing, analysing and reviewing an agile team's performance is important when monitoring productivity. Successful software delivery is only possible if individuals are committed to their work, are provided with the necessary tools and have access to a stable working environment. In addition, individual factors such as knowledge, skills, abilities, personalities and experience should be considered when forming agile teams. Consideration of these factors will result in grouping people that are able to work together and achieve a common goal, which is important in improving productivity. A conceptual framework for agile team productivity was proposed. The discussion of the findings is presented in more detail in this research.

KEY TERMS:

Productivity; Agile; Agile teams; Agility; Metrics; Productivity factors; Performance monitoring; Agile practices; adaptability; Agile methods; Productivity measurement; Scrum; Extreme programming; Agile processes.

DECLARATION

Name: Viola Nzou

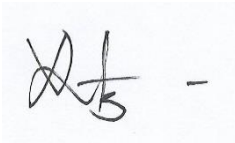
Student number: 4656-394-6

Degree: Master of Science in Computing (98961) (Full Dissertation)

Title of the dissertation as appearing on the copies submitted for examination:

Development of a framework to understand factors that influence productivity in agile teams

I declare that the above dissertation is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.



SIGNATURE

16/10/2017

DATE

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor, Professor Ernest Mnkandla, for his guidance and extraordinary support. His advice, comments, suggestions, reviews and encouragement were beyond measure, and the completion of this research would not have been possible without his support. I truly appreciate his help, guidance and encouragement.

To the management and staff of Synthesis and Digiata Technology Services (Pty) Ltd: I would like to express my sincere appreciation for opening your doors to afford me an opportunity to do my field research. To all the participants who took part in the interviews, thank you so much for your comments, insight and responses. Without you all, this research would not have been completed.

I would like to acknowledge the University of South Africa, School of Computing, academic staff and administrative staff. Thank you for all the workshops, seminars and training that you offered me during the course of my studies. A lot of new things were learnt which improved my research skills. I would also like to thank the University of South Africa student funding for their financial support. All of this dissertation was possible because of your assistance.

Many thanks goes to my husband, Ernest Osaro, and my daughters, Christiana and Stacey, for the indescribable support and love that they have shown me during my research journey. Their love and encouragement made me persevere and work harder to reach greater heights every single day.

My sincere gratitude goes to my mom, Rosemary, and my late dad, Saranavo Hadrian, for teaching me the importance of education from a very young age. Your support and love has kept me going to this day. Many thanks to my brothers and sisters who have always been there for me during my research journey. Your contribution, encouragement and constructive criticism has made me a firmer, harder-working person.

Lastly, I would thank my family, friends, colleagues and everybody for supporting me throughout my research journey. My achievements were made possible with your inspiration and support.

TABLE OF CONTENTS

ABSTRACT	ii
DECLARATION	iv
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	v
LIST OF TABLES	vi
ACRONYMS	vii
Chapter 1: Introduction	1
1.1 Research background	1
1.2 Problem statement	5
1.3 Research context	7
1.4 Research questions	7
1.5 Research objectives	8
1.5.1 Main objective	8
1.5.2 Secondary objectives	8
1.6 Research methodology	9
1.6.1 Research design	9
1.6.2 Research method	10
1.6.3 Data collection methods	10
1.7 Ethical measures	10
1.8 Chapter outline	10
Chapter 2: Literature review and analysis	12
2.1 Introduction	12
2.2 Agile software development models	13
2.3 Differences between agile and traditional paradigms	16
2.4 Agile teams and the concept of agility	19
2.4.1 The concept of agility	20
2.5 Definitions of software productivity	23
2.5.1 Measurement of software productivity	27
2.5.2 Historical software measurement techniques	29
2.5.3 Reasons for measuring software productivity	30
2.5.4 Software measurements and productivity factors	31
2.5.5 Overview of software productivity factors	35

2.6 Productivity in agile teams.....	39
2.7 Summary	40
Chapter 3: Research methodology and design	42
3.1 Introduction	42
3.2 Research methodology	42
3.2.1 <i>The research context</i>	43
3.2.2 <i>The potential research outcome</i>	44
3.2.3 <i>Theoretical framework or proposed conceptual framework</i>	45
3.2.4 <i>The research approach</i>	47
3.2.5 <i>The research philosophy</i>	50
3.2.6 <i>Discussion and rationale for the choice of research strategy</i>	55
3.3 Data collection and analysis	58
3.3.1 Interviews.....	58
3.3.2 Group interviews (focus groups).....	60
3.3.3 Data generation methods for this study in context	61
3.3.4 Analysing textual data	62
3.3.5 Non-textual qualitative data analysis	62
3.4 Research design	63
3.4.1 A model of the research process.....	63
3.4.2 Overview of the research process	65
3.5 Summary	72
Chapter 4: Data collection.....	73
4.1 Introduction	73
4.2 The Interviews.....	74
4.2.1 <i>Selecting the interviewees</i>	75
4.2.2 <i>The interview procedure</i>	76
4.2.3 <i>Conducting interviews</i>	77
4.2.4 <i>The interviews – Process</i>	78
4.3 Data analysis	78
4.3.1 <i>Data reduction</i>	80
4.3.2 <i>Data display</i>	81
4.3.3 <i>Conclusion drawing and verification</i>	82
4.4 Foundation of data collection foundation – Factors that affect agile team productivity	82
4.4.1 <i>Collating agile productivity factors</i>	83
4.4.2 <i>Validating agile productivity factors</i>	85
4.5 Data collection	86

4.5.1 Data collection – Interviews	88
4.5.2 Data collection – Importance of adopting agile methodologies.....	89
4.5.3 Data collection – Factors that influence productivity in agile teams.....	90
4.5.4 Data collection –Definitions and metrics of software productivity	107
4.5.5 Data collection – Agile productivity metrics and performance monitoring	109
4.5.6 Data collection – Monitoring productivity in self-managed teams	111
4.6 Data analysis – Productivity factors.....	113
4.7 Summary	115
Chapter 5: Discussion of findings	116
5.1 Introduction	116
5.2 The results	116
5.2.1 Productivity factors	116
5.2.2 Importance of productivity.....	118
5.2.3 Metrics.....	119
5.2.4 Productivity monitoring	120
5.3 Updated proposed conceptual framework for agile team productivity	121
5.4 Limitations of this research.....	123
5.4.1 Unavailability of participants	123
5.4.2 Lack of available data	123
5.4.3The research strategy.....	123
5.4.4 Researcher’s access	124
5.5 Possible future research.....	124
5.6 Summary	124
Chapter 6: Research summary and conclusion	125
6.1 Introduction	125
6.2 Research summary	125
6.2.1 Overview of the chapters	126
6.3 Objectives achievements summary	127
6.4 Summary of contributions.....	129
6.5 Recommendations for implementation	130
6.6 Future research opportunities.....	130
6.7 Conclusion	131
APPENDIX A - INTERVIEW PROTOCOL	132
APPENDIX B - PARTICIPANT CONSENT FORM.....	134
APPENDIX C - INTERVIEW QUESTIONS	135
APPENDIX D - PAST PAPERS PRESENTED.....	138

APPENDIX E - ETHICS CLEARANCE CERTIFICATE 139
APPENDIX F - LANGUAGE EDITOR CERTIFICATE..... 140
REFERENCES..... 141

LIST OF FIGURES

Figure 2.1 Approaches to new product development	19
Figure 2.2 Triple-P model	25
Figure 2.3 Productivity factors	36
Figure 3.1 The proposed conceptual framework	46
Figure 3.2 Research phases	64
Figure 4.1 Pre-data collection stages	74
Figure 4.2 Team at an early morning daily stand-ups	94
Figure 4.3 Team sprint planning activities	97
Figure 4.4 Developers' work setting (Company X)	99
Figure 4.5 Agile team having an informal discussion	102
Figure 4.6 Developers' work setting (Company Y)	107
Figure 4.7 Burndown chart with summary	110
Figure 4.8 Activity setup rescue time	112
Figure 4.9 Rescue time weekly report	113
Figure 5.1 Updated proposed conceptual framework	122

LIST OF TABLES

Table 1.1 Secondary objectives	8
Table 2.1 Agility definitions and descriptions	21
Table 2.2 Software productivity factors	36
Table 2.3 Summary of factors and their effect on software productivity.....	37
Table 3.1 Secondary objectives	44
Table 3.2 Characteristics of quantitative and qualitative research	49
Table 3.3 Comparison of the three research philosophies	51
Table 3.4 Advantages and disadvantages of interviews	59
Table 3.5 Advantages and disadvantages of interviews	60
Table 3.6 Research stages	65
Table 3.7 Finalising interview questions	71
Table 4.1 Collated agile productivity factors and descriptions	83
Table 4.2 Summary of project and company profiles	86
Table 4.3 Extreme Programming and Scrum practices adopted by the companies	87
Table 4.4 Summary of the effect of productivity factors on the companies under study	114

ACRONYMS

Acronym	Meaning and Definition
IT	<p>“Information Technology”</p> <p>Computing technology which involves developing, maintaining and using computers and software for information processing and distribution.</p>
IEEE	<p>“Institute of Electrical and Electronics Engineers”</p> <p>A non-profit organisation for educational and scientific research.</p>
std	<p>“Standard”</p> <p>A unit of measuring quantity.</p>
LOC	<p>“Lines of Code”</p> <p>A software metric for measuring computer program size by counting non-commentary lines of program source code.</p>
FP	<p>“Function Points”</p> <p>A unit measure of software.</p>
PC	<p>“Personal Computer”</p> <p>A multi-purpose microcomputer designed to be used by one individual at a time.</p>
DDJ	<p>“Doctor Dobb’s Journal”</p> <p>A monthly journal published by United Business Media in the United States covering computer programmer topics.</p>
%	<p>“Percentage”</p> <p>A ratio or number expressed as a fraction of one hundred.</p>
XP	<p>“Extreme Programming”</p> <p>An agile discipline of program development that emphasises simplicity, business results, feedback, communication, respect and courage.</p>
DSDM	<p>“Dynamics Systems Development Method”</p> <p>An agile-approach project delivery framework.</p>
SDLC	<p>“System Development Life Cycle”</p> <p>The sequence of stages involved in the software development of an information system, from planning through to deployment.</p>
P-model	<p>Triple-P Model (Performance, Profitability and Productivity)</p>

	A model that shows the relationship between performance, profitability, and productivity.
SLOC	“Source lines of code” A sizing metric for measuring the size of the source code.
EVMS	“Agile Earned Value Metrics” A project management technique that is used to monitor and measure agile team productivity.
KW	“Knowledge Worker” An individual with subject matter expertise, who is able to share his/her knowledge and is part of the program analysis process.
KLOC	“Thousands of lines of code” A traditional way of measuring source code.
<i>Inp</i>	“Input items” The totality of the resources used, for example; labour, tools, machines; etc.
<i>Out</i>	“Output items” The number of produced items.
<i>maf</i>	“Master file” A records file that is permanent, updated periodically and is the main source of data.
<i>inf</i>	“Interfaces” A shared boundary that allow users to exchange and share information.
COCOMO	“Cost Constructive Model” A software cost estimation model developed by Barry W. Boehm which uses a regression formula based on historical projects data.
JIRA	“JIRA” A tracking product developed by Atlassian, which provides issue and bug tacking, as well as project management functions.
E-mails	“Electronic Mails” A method of exchanging computer-stored messages over computer networks.
TL1 & TL2	“Team Leader 1 or Team Leader 2”

	A person who is responsible for leading the team, obtaining resources for the team, facilitating the team's work and ensuring that the team is protected from problems.
P1 & P2	"Programmer 1 or Programmer 2" An individual who is responsible for coding or development of software.
PO1	"Product Owner 1" (Also called an On-site customer.) A person who is responsible for a team's prioritised work item list, for timely decision making and for providing the needed information in a timely manner.
PM 1	"Project Manager 1" A person who is responsible for project management activities from project planning through to project execution.
LP 1 & LP2	"Lead Programmer 1 or Lead Programmer 2" A person who is a senior programmer and has the leading role when coding in an agile team.

Chapter 1: Introduction

1.1 Research background

Software productivity is an important subject in software development nowadays. Business requests for reduced marketing time, while trying to keep up with good software quality; in addition, software development organisations seek new techniques and strategies to increase productivity in their software development operations. (Trendowicz & Münch 2009:185). About thirty years ago, Boehm (1987:1) expressed the opinion that the rate of computer hardware productivity kept increasing rapidly, while software productivity stayed on the same level with minimal increase. Hardware productivity has increased drastically, while software projects productivity has come to a standstill, basing productivity on the code using the unchanged old rate of lines of code delivered per man-hour.

Finding ways to manage productivity continues to create a challenge for software projects till this day. Systems have been designed and tested in the manufacturing sectors for measuring productivity, but the information technology industry lingers behind in designing metrics for estimating the software end product and predicting the work required to complete each software project (Dalcher 2006:93). According to Zimmermann (2017:1), “there is an ever-growing demand of software being built and a shortage of software developers to satisfy this demand”. To address this challenge, researchers and the industry need to look into understanding what productivity is in software development, and to find ways to improve the productivity of a team as well as of individual developers. Software productivity management in the IT segment is significant as a fundamental variable for knowledge workers (Ramirez & Nembhard 2004:602-628) and as a pivotal angle for making decisions in the worldwide business sector (Ross & Ernstberger 2006:30–32). Hence, software productivity is an important concept and much talked about in today’s projects and organisations.

Productivity, in economic standard terms, is expressed as the ratio of the measure of products or services delivered, to the measure of labour or cost incorporated in producing the final product (Jones 1996:445-460). A presumption that precedes this is that software productivity is

expressed as the ratio of the measure of software delivered to the labour and cost of producing it (Jones 1996:445–460). Usually, the productivity of mechanical procedures was measured as the ratio of output units yield divided by input units (National Bureau of Economic Research 1961:112). This point of view was moved into the software development setting and is generally characterised as defining productivity (IEEE Std 1045, 1992) or efficiency (van der Pohl & Schach 1983:187-191).

As noticed in an international survey conducted by the Fraunhofer Institute for Experimental Software Engineering in 2006, 80 % of software organisations adopt this industrial point of view of productivity in their software projects, namely, that input consists of the effort used to deliver outputs. Such organisations presume that software productivity measurement is the same as measuring any other types of productivity. Hitherto, software development procedures appear to have been more challenging than other industrial production (Abdel-Hamid 1996:43-52; Angkasaputra *et al* 2005:83-92; and Briand *et al* 1998: 65-117). This is basically due to the fact that software organisations normally develop new products instead of manufacturing the same product repeatedly. Additionally, the human factor is involved in software development, which normally has compelling instabilities from the onset. This prompts numerous difficulties in defining “software productivity”.

Productivity involves not just the efficiency (amount of output divided by amount of input) of creating a product, but also the rate at which the output is produced, given a certain input (Capers 1996:94-103). Productivity in software development is the ratio of quality software delivered, to the labour and cost of producing the end product (Capers 1996:94-103). The tools used to evaluate or measure productivity in software development consider the usefulness conveyed to the client, program complexity, and the effort and time included. Mills (1983:57) defined software productivity as the ratio of functional values of produced software to the labour and expense involved in producing the end product. The definition given permits the measurement of productivity in light of the estimation of aftereffects to end user, which can be reasonable than constructing productivity results in light of lines of code (LOC) (Mills 1983:57-63).

Software productivity has lagged behind due to deficiencies in its measurement (Scacchi 1994; Drucker 1999 & Petersen 2011), for instance, by means of Function Points (FP) and Lines of Code (LOC), and to a limited extent due to the way that quicker, more effective PCs can give faster execution without actual software productivity gains.

Agile software development stresses working in a team of small groups as an important contributing factor in producing high software quality, with information-sharing among programmers (Consas, *et al* 2007:54). Cockburn and Highsmith (2001:131-133), conducted a survey according to which agile methodologies were rated higher compared to others as far as morale is concerned. According to the 2007 IT Project Success Rates Survey Results presented by Ambler (2007) in Dr. Dobb's Journal (DDJ) in October 2007, the survey results demonstrated that agile work was superior to traditional methods. The survey demonstrated that when individuals characterise accomplishment in their own particular terms, agile projects had a 72 % achievement rate, contrasted with the 63 % for the traditional approach and 43 % for offshoring.

The DDJ 2008 Agile Adoption Survey demonstrated that the involvement of individuals with agile software development was exceptionally positive, and that embracing agile strategies seems, by all accounts, to be generally safe in practice. The July 2010 State of the IT Union survey investigated the issue of team size and agile approaches, showing an improvement over traditional approaches despite any team size. The 2008 IT Project Success Rates Survey investigated the issue of geographic dissemination and agile approaches, which did well or better despite the level of team circulation. In 2013, in an IT Project Success Survey conducted by Scott Amber & Associates, they figured out that agile, lean and iterative strategies were thought to be better by and large than traditional and ad-hoc strategies.

Agile methods are seen as a solid contributor to high morale in software-producing teams, and there is a strong association of agile practices with the idea of a positive 'group atmosphere' that can add to superior performance (Pressman & Maxim 2015:692).

According to Boehm (1987:43-58) and Trendowicz (2009:185-214), low costs and time-to-market have turned into real drivers of software productivity improvement. In spite of the

endeavours to expand software productivity, delivery in time and within the budgeted amount is still notably low; and there is not much indication of productivity improvement over time. In spite of the fact that studies on productivity have been done for the past three decades, it still continues to be a controversial subject (Trendowicz & Münch 2009:185-214), Scacchi 1995:37-70, Eickelmann 2001:67-70 and Riddle & Fairley 2012:16). Firstly, there are several concepts included in the definition of productivity, for example, performance, effectiveness, and efficiency, creating incorrect assumptions, and word overload. Besides, traditionally, the measurement of software productivity was expressed as the ratio of output (for instance, actualised features, LOC, and FP) to input (for example time and effort). What's more, developing software is mental work involving the creation of knowledge or, in any case, the use of knowledge as a predominant piece of labour (Ramirez & Nembhard, 2004:602-628).

A good deal of the literature dealing with software development methodologies contains references to socio-mental issues, for example, inner self, prosperity, control and team conflict (Bullock, Weinberg & Benesh 2001:37-43; DeMarco & Lister 1999; Weinberg 1971). Indeed, even in this regard there is an absence of essential research into the productivity of each team member in an agile team (or whatever other programming team) (Whitworth & Biddle 2007:62-69). The agile and software engineering literature was observed to be predominantly written from the management and engineering points of view, concerning the practicalities of software development, software development processes management, and the obstacles of making everything work within the business setting (Whitworth & Biddle 2007:62-69).

The purpose of this study is to develop a framework for understanding the factors influencing software productivity, in an up-to-date review of software development. A review of the concept of productivity, productivity processes and parameters through literature review shall be carried out. The importance of software productivity for organisations, factors influencing productivity of agile teams, and an investigation of attempts to identify threats to productivity of agile teams shall be investigated through case studies and literature studies.

1.2 Problem statement

According to Pressman and Maxim (2015:66), agile software engineering comprises a philosophy and guidelines used in software development. The agile philosophy inspires developers to produce software that will satisfy the customer and the software is delivered early in small portions; the whole process is characterised by profoundly energetic project teams, casual routines, marginal software end products, and simplicity in software development. According to Pressman and Maxim, the guidelines for agile development lay emphasis on delivery over analysis and design, and on active constant communication between programmers and customer. Agile software development brings a sensible different option to traditional software development for specific types and classes of software. Zimmermann (2017:1) asserts that there have been significant changes in software development over the past decades due to the introduction of agile methodologies.

Agile methodologies, including Scrum (Schwaber & Beedle 2001) and Extreme Programming (Beck & Andries 2004), advanced as ways to deal with improving the software development process, possibly prompting improved productivity. These agile methods aim to improve the time to develop the software, and to tackle the unavoidable changes coming from market changes (Karströmand & Runevon 2006 and Pikkarainen *et al* 2008). The use of agile methodologies is therefore aimed at enhancing the productivity of software development. It appears that organisations consequently expect that, just by going agile, or any other related methodologies like Scrum, Kanban or comparative methodologies, more software will be built in less time. Hence, it is postulated that understanding the factors that influence productivity may assist in determining areas on which to focus a company's financial resources and management efforts from a practical point of view, and probably where to centre research effort from the academic viewpoint (Rasch and Tosi 1992).

Substantial research has been focused on finding factors that significantly affect software development productivity (Trendowicz & Münch 2009 and Wagner and Ruhe 2008). Productivity is not just only influenced by simple application of any methodology, but also by the organisation's culture, the amount of requirements change, the expertise and experience of

individuals in the teams, extraneous factors such as external parties or suppliers, or the complexity of the application background and architecture. Constantly assessing factors that influence productivity is critical, as these factors are bound to change at any time with the introduction of new software development practices (Petersen 2011). However, from our own understanding, there is little study in literature that has explored the main factors affecting the productivity of agile teams (Trendowicz 2009; Zimmermann 2017)

In addition, empirical evidence points out the lack of studies on agile productivity measurement (Dybã and Dingsoyr 2008; Petersen 2011), as well as team productivity measurement (Priers-Heje and Commisso 2010). To figure out whether productivity is improving, agile teams and projects should be measured and contrasted with traditional teams and projects. Throughout the research, the researcher should investigate how developers define productivity and metrics in agile teams. Thus, empirical evidence studies on productivity definitions and metrics may answer questions on how to improve agile team management, as well as how to monitor productivity in a highly adaptive scenario.

The continued and increasing demand for software development and maintenance has made productivity an important issue (MyNatt 1990). According to Sidler (2002), the productivity concept in software development cannot be only a “theoretical abstract”. Sidler (2002) stated that productivity is an important concept in the software development process. Therefore, getting to understand productivity in software engineering is critical in system analysis when considering that better systems analysis improves productivity in software development and productivity in software development is a success measure of systems analysis.

In summary, the overall problem addressed by this research is:

Addressing issues that come out of software development methodology and processes that affect certain parameters (quantity and quality) in determining productivity. The research also focuses on gaps in measuring productivity, productivity metrics and indicators.

1.3 Research context

This research will use qualitative data from companies in South Africa (limited to Gauteng Province and Western Cape Province) that adopted agile methodologies and have been using these methodologies for more than three years. In South Africa, agile is a new software development tool and most of the companies are still in the pilot phase in implementing agile methods in their software development (itweb; iweek; Noruwana & Tanner 2012). The selection criteria incorporated the following: (i) organisations utilising agile methods Scrum (Schwaber and Beedle 2001), and (XP) (Beck et al 2004), for not less than four years; (ii) organisations in diverse business portions, geographical locations, different in size, and having different organisational structures and cultures; (iii) agile software projects with four or more core programmers.

The research motivation, therefore, is to better understand the factors that influence software productivity, i.e. ways in which software productivity can be defined and measured in agile teams.

1.4 Research questions

RQ1. What are the factors influencing the productivity of agile teams and how do these factors affect team productivity from the team point of view?

SubRQ1.1. Which agile practices have an effect on a team's productivity?

RQ2. What is the importance of productivity on companies that adopt agile methods?

SubRQ2.1. How would they define software productivity?

RQ3. What are suitable metrics for determining agile team productivity?

SubRQ3.1. In scenarios where such metrics have negative effects, are there adjustments that could be made to promote productivity?

RQ4. How should productivity factors in agile teams be monitored, considering agility and adaptability?

SubRQ4.1. How do agile metrics relate to productivity metrics?

1.5 Research objectives

The managing of software development productivity and software quality are the key issues in software development companies, mostly because companies need to lower their development costs and reduce time spent marketing their products (Beck 2004); (Trendowicz 2009). To control productivity adequately, it is vital to recognise the most significant problems and create procedures to adapt them. Agile methods, including Scrum (Schwaber & Beedle 2001), and Extreme Programming (Beck 2004), advanced as a critical way to deal or improve software development processes, possibly prompting better productivity. They lessen the time to develop the software and tackle inescapable changes coming from market complexities (Karlström & Runeson 2006).

1.5.1 Main objective

The main objective of this research is to develop a framework that gives a better understanding of the factors that influence agile team productivity, the importance of productivity in agile teams and ways in which the agile metrics can support productivity metrics.

1.5.2 Secondary objectives

The following are the identified secondary objectives:

Table 1.1 Secondary objectives

Secondary objective	Methods of implementation
1) To define software productivity and finding out its importance in companies adopting agile methods.	<ul style="list-style-type: none">• Document analysis through focused literature search.• Semi-structured interviews with identified companies using agile methods in South Africa.

Secondary objective	Methods of implementation
2) To determine factors that influence agile team productivity, and their actual influence.	<ul style="list-style-type: none"> • Reviews of related literature. • Observation of agile teams. • Reviews of publications.
3) To determine which metrics are suitable for measuring an agile team's productivity.	<ul style="list-style-type: none"> • Reviews of related literature. • Reviews of publications.
4) To find out ways of monitoring productivity factors in agile teams, considering agility and adaptability.	<ul style="list-style-type: none"> • Reviews of related literature. • Reviews of publications.
5) To explore agility properties to better understand how to define and evaluate software productivity in a highly flexible, adaptive and response environment.	<ul style="list-style-type: none"> • Reviews of related literature. • Reviews of publications. • Semi-structured interviews.

1.6 Research methodology

Oates (2009:5) defined research as creating new knowledge by following appropriate procedures, to satisfy people who will use the research. Therefore, research involves a type of thinking that is done to come up with results. The research methodology involves strategies and data generation methods which highlights the research process chosen to conduct this study (Oates 2009:34). The motivation for the choice of the research strategy, and the data generation method, are presented in more detail in Chapter 3.

1.6.1 Research design

For this study, research design is a planned research approach built on the justification of the selected research method found in the section on research methodology. The research design answers the 'how' questions whilst the methodology answers 'why' and 'what' questions. The research design will be discussed in detail in Chapter 3, describing the processes of collecting and analysing data.

1.6.2 Research method

Empirical studies may apply different research methods. This research will be conducted using a qualitative research method, namely case studies, in order to obtain a good understanding of the problem that is being investigated. The selected research method is appropriate in answering the research questions focused on “what” and “how” (Yin 2008).

The researcher intends to explore which productivity concepts are practised by agile teams in their software development projects. The researcher likewise plans to ascertain which factors have an effect on the productivity of agile teams, and to investigate whether such agile practices have an effect on productivity from the team’s point of view. The researcher will use case studies, to answer the research questions.

1.6.3 Data collection methods

A number of sub-methods will be utilised in conducting the research; they are further described in detail in Chapter 3.

1.7 Ethical measures

The researcher has to sign a confidentiality agreement with any organisation that participates in the case studies; this process is essential to establish a formal understanding between the researcher and the organisations involved, for data confidentiality to be maintained. All information or data collected will be strictly confidential and shall not in any way be disclosed to any third party but be used solely for research purpose. The researcher will abide by the University of South Africa ethics policy as outlined in the ethics clearance issued by the University to conduct field work.

1.8 Chapter outline

Chapter 1: Introduction and background

Chapter 1 presents the introduction, problem statement, research questions, research objectives, research methodology and current theories.

Chapter 2: Literature review

Chapter 2 presents a comprehensive theoretical background of the principles of agile methods, the most frequently claimed benefits, the concept of agility, and some paradoxes in the definition of agile software productivity.

Chapter 3: Research methodology

Chapter 3 presents research methods and research design

Chapter 4: Data collection methods and analysis

Chapter 4 discusses data collection methods and analysis, merits and demerits of the methods adopted in the dissertation: case studies, interviews, etc.

Chapter 5: Findings

Chapter 5 provides a detailed discussion of the findings. The chapter describes the research findings for RQ1 and RQ2, discusses the researcher's findings in comparison with the existing literature. It further describes the research findings for RQ3 and RQ4, the researcher's results, discussion, and limitations.

Chapter 6: Conclusion

Chapter 6 concludes the research by presenting an overview of the dissertation and proposing future work.

Chapter 2: Literature review and analysis

The previous chapter outlined the background of the research study that is to develop a framework for understanding the factors that influence software productivity in agile teams. Chapter 2 follows with a literature review of agile development methodologies, the agility concept, factors that influence team productivity and empirical studies of productivity. The issues noted above lead to the investigating, defining and understanding of the research problem relevant to developing the sought framework.

2.1 Introduction

In the current economy, changes in market conditions are always high. End-user and customer need change, and competition in the software development industry becomes intense. Software experts should maintain their “agility” in software engineering in order for them to be able to adapt to maneuverable, adaptive, lean processes that accommodate today’s business needs (Pressman & Maxim 2015:708).

The philosophy of agility in software engineering lays emphasis on four key issues which are: (1) how important “self-organising” teams are, since they have some level of control over the work they carry out; (2) how team members, practitioners and customers communicate and collaborate among themselves; (3) recognising that change is good and signifies a prospect and (4) focusing on continuous software delivery to the satisfaction of the consumer. Agile process methods aim at addressing these issues which are prevalent in the traditional software models such as Waterfall (a sequential, non-iterative traditional software design model) (Pressman & Maxim 2015:708).

Despite attempts made in earlier years to improve the software development by concentrating on finding better ways of defining and documenting requirements, analysing the requirements, and delivering a working software on time and within budget, this was never achieved. While many approaches to software development were considered as iterative models, they were

unable to address the needs of managing rapidly changing requirements effectively, nor of shortening the software delivery time (Avison & Fitzgerald 2006:134 -135).

Many of the software development processes in the 1980s and 1990s were criticized as being slow, bureaucratic, and over-grouped. In response to these heavy-handed software development methods, a number of dissatisfied software developers proposed agile methods in the 1990s. This group of seventeen software developers gathered together and produced a “Manifesto for Agile Software Development” (Beck et al, 2001). It enabled the software development teams to focus on delivering the working software instead of focusing on software design and documentation (Sommerville, 2007:396). No specific life-cycle model was prescribed by the Agile Alliance, but it rather came up with a group of underlying principles which were common to “their individual approaches to software development” (Schach 2007:58).

Agile processes differed from other software development models by laying less emphasis on analysis and design. Working software is considered quite essential, unlike detailed documentation, so implementation has to start earlier in the life-cycle of the software. Responsiveness to changes and collaboration with the customer is another major goal in agile development (Schach 2007:58). Agile processes are considered to be light methods, or lean methods (Pressman & Maxim 2015:67). In principle, agile methodologies were established to try and overcome observed and real weaknesses of traditional software development.

This chapter will give a brief outline the history of agile software development, mainly focusing on the differences between traditional development methods and agile methods. It will also include the issue of agility and how it relates to productivity in agile teams. The history of productivity measures will also be noted, together with their relevance to agile software development.

2.2 Agile software development models

Even though the fundamental concepts that guide agile software development have been in existence for many years, it was only more than two decades ago that such concepts were

considered as a “movement” (Pressman & Maxim 2015:67). The history of software development approaches such as the ‘agile’ one goes back to before the Agile Manifesto (Avison & Fitzgerald, 2007:134). Highsmith (2002:6) stated that the history extended back more than ten to fifteen years. The agile approach appeared in different variants but always from the same starting point: to try and overcome the perceived and actual inadequacies of the traditional approach to software development (Avison & Fitzgerald, 2007:134).

According to Schach (2007:59), “agile processes have been successfully used on a number of small-scale projects, but, they have not yet been used widely enough to determine whether this approach will fulfil its early promise.” According to Sommerville (2007:398), agile methods are most suitable for development of software for small to medium-sized computer applications and business systems. Agile methods generally depend on an iterative approach to specifying, developing and delivering software, and were developed initially to support business applications with rapidly changing system requirements during the development stage (Sommerville, 2007:396). Agile methodologies are designed to release working software frequently to customers, enabling them to suggest new or any changes in requirements to be added in the later iterations, enabling high-quality software to be produced in a profitable way (Sommerville 2007:11).

There are quite a few agile methods, which were developed to date with the intention of producing cost-effective software as quickly as possible. The best-recognised agile method is probably XP (Extreme programming) (Beck 1999:72; Beck 2000:2). Other agile approaches such as Crystal (Cockburn 2001:4), Scrum (Schwaber & Beedle 2001:2), DSDM (Stapleton 1997:3), Feature Driven Development (Palmer & Felsing 2002:135) and Adaptive Software Development (Highsmith, 2000:85) were also successful; this led to some getting integrated into traditional development methodologies, which resulted in the notion of agile modelling (Ambler & Jeffries 2002:87), and in instantiating agile frameworks such as Rational Unified Process (RUP) (Larman 2002:31).

Despite the fact that all these agile methods base their software development on incremental value delivery, they recommend a variety of procedures to achieve this. Yet agile methods share

a set of values and principles. Unlike conventional methods, agile methods depend on humans and their creativity to deal with unpredictable changes, rather than on rigidly plan-based processes (Cockburn 2002:72).

The agile approach intends to attain high software productivity and high-quality software delivery, in a cost-effective manner, enabling it to respond to business challenges, which demands high quality, high performance and high development speed in delivering their goods and services (Pressman & Maxim 2015). Therefore, agile methods are considered to be highly value based.

The following are the frequently claimed benefits of agile methods (Highsmith 2001, 2002 & 2009; Demarco & Boehm 2002:90-92):

- High degree of stakeholder engagement.
- High level of transparency between the agile team and the customer.
- Delivering new features quickly and with a high predictability level.
- Predictable cost estimates and schedules.
- Focus on providing the most business value.
- Provide opportunities for changes to be introduced within a few weeks.
- User focused acceptance.
- Improved quality of software through conducting testing and reviews during each iteration.

However, all development methods have their own limits, and agile methods have their own disadvantages (Sommerville 2007:398; Highsmith 2002:197 and Kettunen 2009:33). Despite their limitations, agile process models address each of the problems arising in software development environments, by providing sustainable software development, shorter times to market, and harnessing changing requirements and the changing environment (Pressman & Maxim 2015:68; Baskerville et al 2006:14).

The aim of the above discussion is to answer the research question based on the importance of productivity for companies that adopt agile methods. Companies worldwide are adopting agile

methods because they claim that agile methods allow them to manage changing requirements, accelerate time to market and improve productivity.

2.3 Differences between agile and traditional paradigms

A group of software development methods called agile software development are considered to have solutions which advance through teamwork and “self-organising” (Collier 2001:53), cross-functional teams. Agile software development encourages continuous software delivery, adaptive planning, quick responses to change, continuous software improvement and the production of working software (Agile Alliance, April 2016).

Most agile methods allocate tasks into small “sprints” with short-term planning, and no direct long-term plan is involved. Sprints are short time cycles called ‘timeboxing’ (Jalote et al 2004:117-118) that generally take one week to a month to complete. The cross-functional team is involved in each iteration working in all functions, e.g. sprint planning, scope analysis, design, coding, unit and acceptance testing. A working product is demonstrated to end users when each iteration ends. This allows for risk to be minimised and enables the project to quickly adapt to changes (Sommerville 2007:412). Although each sprint release might not be enough to add functionality that can guarantee market release, the aim is to have at least a software release with minimal bugs at the end of each sprint. (Beck 1999:70-77)

Traditional paradigms are plan-driven and have discrete development phases, starting with analysis and requirements documentation, followed by system design, then system development and testing. In fact, each stage must be finalised to make it possible to proceed to the following stage. (Sommerville 2007:804). Because of these “heavyweight” aspects, the paradigms of this approach were then termed heavyweight. Most software experts found the heavyweight process frustrating, time consuming and posing difficulties if there were any changes in requirements during the early stages of software development. In reaction to this, several software experts started independently developing practices and methods that could allow quick response to the change that most software development organisations were experiencing. The practices and

methods developed are based on a technique introduced in 1975 and then called iterative development; now it is called agile methodology (Awad 2005:16).

Agile methodologies have evolved from adaptive software development, whilst traditional (System Development Life Cycle) methods, for example, the waterfall, use a predictive approach to system development. In traditional methodologies, a team works with a well-defined plan with detailed tasks and activities that should occur in sequence, and each phase should be completed before moving on to the next phase. Checklists and forms are used as guidelines in the entire life cycle of the product (Boehm & Turner 2004:195). Traditional methodologies solely rely on comprehensive planning and requirements analysis at the beginning of the product life cycle. To make any changes to the requirements, everything has to go through proper change management and prioritisation processes (Nogueira et al 2000:3). These prescriptive models do not seem to be appropriate for a software development industry that thrives on change. Hence, agile methods came as an alternative to traditional methods and suggested a “revolutionary change” (Pressman and Maxim 2015:67).

An agile development has the ability to respond quickly to unpredictability by relying on people who build computer software rather than on prescriptive process models (Cockburn 2002:72). An agile software process is characterised by iterative development, rapid development and frequent delivery, and is initially designed to support applications in a business where there is rapid change in requirements during software development stages. There should be frequent delivery of software increments to keep up with change (Pressman & Maxim 2015:70). The development team works with the customer, enabling the evaluation of software under development on a regular basis, with the customer giving the required feedback to team members, and influencing the decision-making to accommodate the feedback.

Agile methods encourage knowledge creation and transfer, by involving the customer intensively during software development and by frequent feedback from customers through direct oral communication, which allows efficient individual knowledge transfer. Team members like to face changing requirements, to reflect on how to become more effective, and to enhance their knowledge and skills through a “learning-by-doing” process (Garud & Kumarasamy 2005:22).

Documentation is kept at minimum in agile development as it is perceived to be “non-creative” as well as “non-productive”, resulting in static documentation which does not reflect the actual work and may never be of use in the future after its creation (Agile Manifesto 2005). The difference between traditional (plan-driven) and agile (adaptive) is then that the agile approach recognises that achieving a planned milestone doesn’t automatically mean client success is achieved (Mellor 2005:18). With agile development, the customer is involved in the entire software development process, to ensure adherence to system requirements as well as customer satisfaction, and to gain a competitive advantage. Frequent “agile” iteration gives opportunities for quick feedback on product adherence to specifications (Karlström et al 2005:43). Programmer commitment, motivation, team spirit, and morale are essential success elements (Siakas et al 2003:169; Abrahamsson 2002:10). Hence, agile methods are a current alternative to traditional software development methods (Oz 2008:434).

Figure 2.1 illustrates the difference between the traditional model of new product development and a more flexible approach.

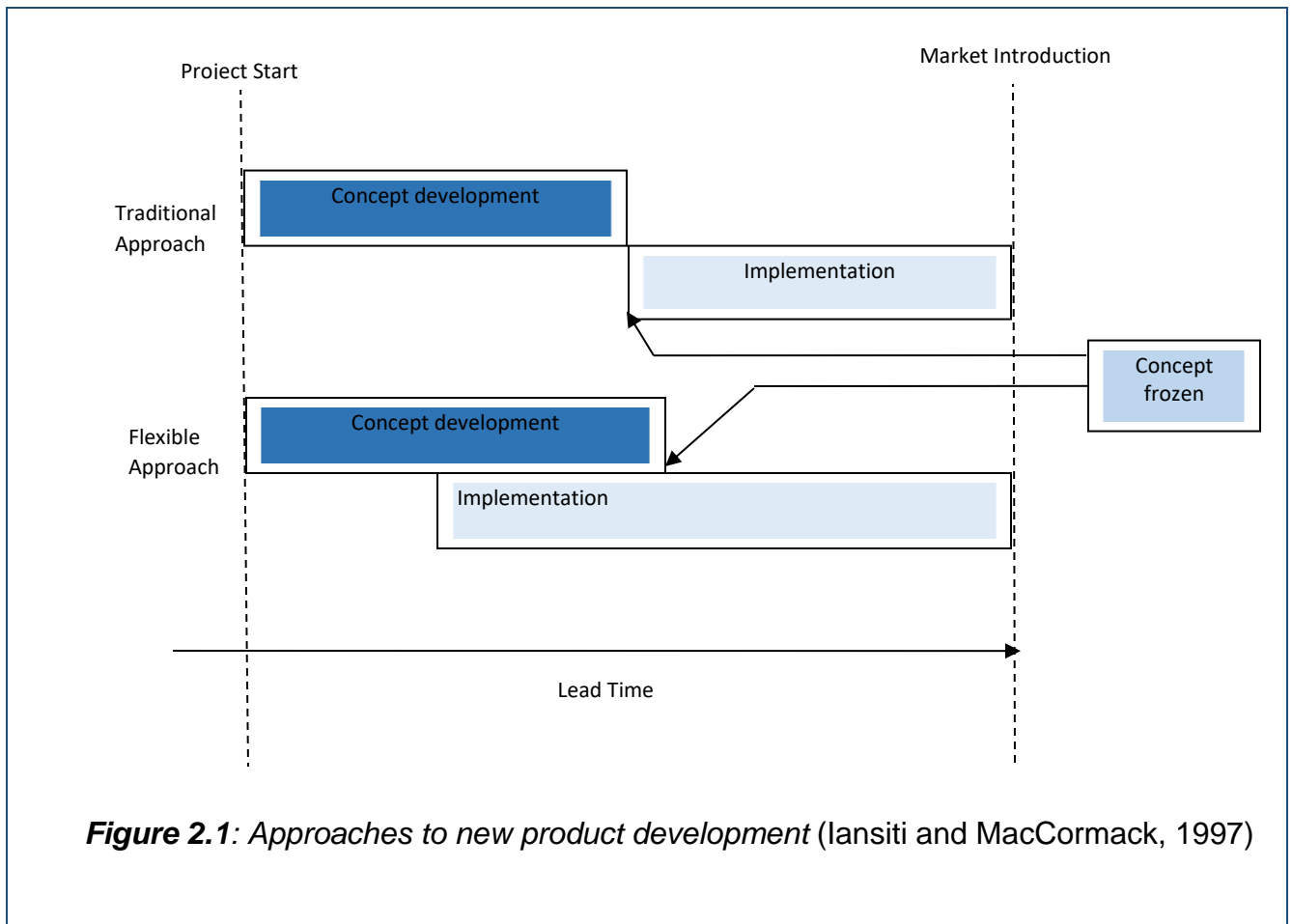


Figure 2.1: Approaches to new product development (Iansiti and MacCormack, 1997)

The focus of agile approaches emphasises iteration, short life cycles, simplicity, emergence, people orientation, teamwork, risk acceptance and communication (Pressman & Maxim 2015). Hence many companies adopt agile methods rather than traditional methods. Companies not only adopt agile methods expecting productivity improvement, but also perceive productivity as one of the most important benefits of agile methods (Oz 2008). This segment aims to answer the research question on the importance of productivity on companies that adopt agile methods as opposed to traditional methods.

2.4 Agile teams and the agility concept

Many software organisations advocate agile software development as a replacement for traditional software development, so that it can deal with problems encountered by software

project work. The agile philosophy lays emphasis on satisfying the customer and delivering software early, on project team members that are highly motivated, on no working procedures, less software engineering work products, and overall simplified development. (Pressman & Maxim 2015: 68 - 72).

The small dedicated project team, known as an agile team, embraces many of the characteristics of an effective software team and avoids unnecessary processes that create problems. In addition, the agile philosophy emphasises competency of individuals, with team collaboration as all-important to achieve project objectives.

Ivar Jacobson (2002:18 – 24) provides a useful discussion:

“An agile team is a nimble team able to approximately respond to changes.” Changes are what software development is very much about. Changes in building the software, grouping of team members, new technology, and making changes to everything that has effect on a product. Support of changes has to be incorporated in everything in software development; it needs to be embraced because it is the core of software.

An agile team can be defined as a “self-organising” team that has liberty to independently plan and make decisions on technical issues... ...A self-organising team need not to maintain the same team structure, rather, it uses principles of “Constantine’s open, synchronous and paradigms” (Pressman & Maxim 2015:68). Agile teams acknowledge that software is developed by different people who work together as a team, and the skills of these individuals, and their ability to work together are important for the project to succeed. The discussion of the agility concept will assist in answering the research question how productivity factors in agile teams should be monitored, considering agility and adaptability.

2.4.1 The concept of agility

The foundation of the agility concept was first introduced by the Japanese automobile manufacturers (Womack et al. 1990:2). Agility is now today’s buzzword used describe modern computer software models (Jacobson 2002:18).

Even though in the literature there is not yet a well-established definition of software development agility, a common theme drawn from its various definitions and descriptions is that preponderance of change is the key driver of agility (Pressman and Maxim 2015:68).

The following are various definitions of agility related to agile software development:-

Table 2.1: Agility: definitions and descriptions

Scholar	Definitions/ Descriptions
<i>Maskell (1999:5)</i>	Defined agility as: <i>the ability to prosper and succeed in an environment dominated by unpredictable and continuous change. Agility is more than just accommodating change, rather, acknowledging the opportunities incarnated within the unpredictable environment.</i>
<i>Jacobson (2002:18)</i>	Describes agility as: <i>today’s buzzword used to describe modern computer software models. Everybody has become agile.</i>
<i>Cockburn (2002)</i>	Defines agility as: <i>being competent and maneuverable; using simple but adequate rules of project management and applying well-defined communication rules.</i>
<i>Conboy & Fitzgerald (2004)</i>	<i>Agility is when an organisation is continuously ready to reactively or proactively, inherently or rapidly, embrace change through simplistic, economical, high quality components, and relate to the environment.</i>
<i>Goldman et al (2005)</i>	<i>Agility is vigorous, content-specific, actively embracing change, and progress-oriented.</i>

Scholar	Definitions/ Descriptions
<i>IEEE (2007)</i>	<i>Agility is the capability to accommodate unknown changes in requirements even in the late stages of software development (it can be until the start of the last iterative development cycle of the release).</i>
<i>Highsmith (2010:5)</i>	<i>Agility is the ability to react and respond to change quickly so as to profit in the demanding business environment. Agility is therefore the ability to balance between stability and flexibility.</i>
<i>Pressman and Maxim (2015:67)</i>	<i>Agility goes beyond only effectively responding to change. It embraces the philosophy espoused in the Manifesto for Agile Software Development. It supports team structures and attitudes, making communication to be more simple; it emphasises frequent software delivery, it adopts the customer as part of the development team; it recognises that planning in a turbulent business world has limitations and that project planning should be more flexible.</i>

Agile development methods go beyond “lean and iterative” techniques. These methods promote continuous innovation, product adaptability, improved return on investment, adaptability of people and processes, and concentrate on dependable products to sustain business growth (Highsmith 2010:19). Proponents of agility argue that agile methodologies are successful because they are relevant to the new ways of thinking and the general software development approach. Many researchers assert that agility is based upon several assumptions, principles, and skills which are worth doing research on (Boehm & Turner 2006:27; Highsmith 2010:6; Chan & Thong 2009:803; Cobb 2011:191).

Considering all the above definitions of agility, it becomes clear that agility can equally be adapted by any software development process (Pressman & Maxim 2015:68). However, for the

present research a nominal conceptual theory must be established that enables one to interpret agility in other software development dimensions, such as productivity. The aim is to explore agility properties to better understand how to define and evaluate software productivity in a highly flexible, adaptive and response environment. One also has to find out if there is any common dimension between software development agility and software development productivity. Abrahamsson et al (2009:281) acknowledge that an understanding of what constitutes agility, is still lacking, due to the shortcomings in agile systems development research. Regarding agility and adaptability, one needs to investigate the importance of productivity for companies adopting agile methodologies in South Africa.

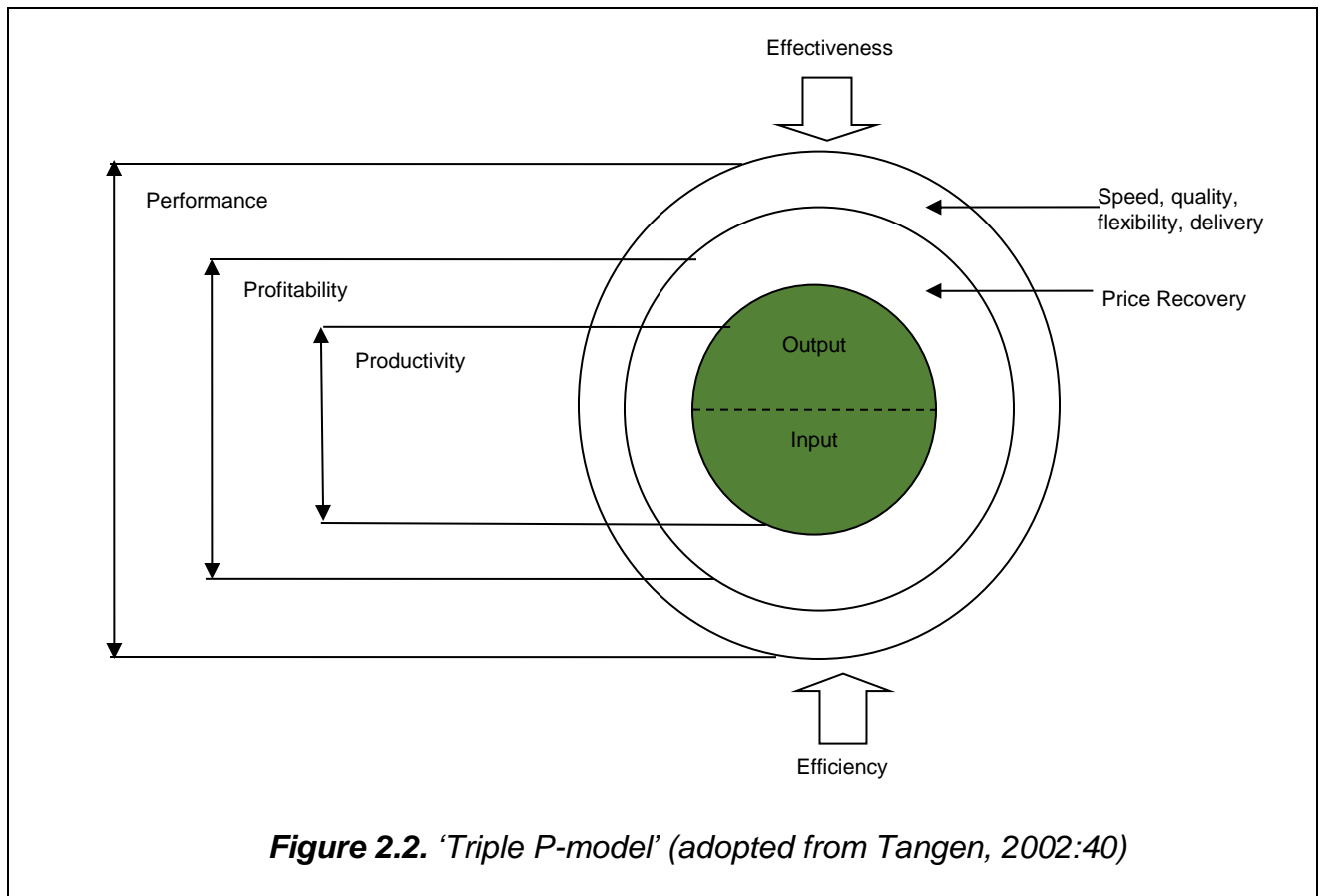
2.5 Definitions of software productivity

Jones (1996:47) defined productivity as the ratio of goods and services produced to labour and expense used to produce them. It should then follow that software productivity is the amount of produced software divided by the labour and cost of producing the software. This theory appears to have a simple logic, but in reality turns out to be a matter of debate. Mills (1983:265) gave a broader definition which states that software is more than a computer program; it also encompasses related procedures and documentation that are associated with the program.

Although software productivity has been researched comprehensively in software engineering, it is surrounded by controversy as to how software practitioners define it in a more understandable way. In his abstract, Tangen (2002:18) stated that even though the productivity concept is a widely discussed topic, it is often poorly understood and vaguely defined. Zimmermann (2017:1) acknowledges that “a substantial amount of work has examined the meaning of software productivity over the past four decades”, but “much of the work introduces particular definitions of productivity, or is focused on specific tools and approaches for improving productivity”. The definition of productivity involves many concepts, including efficiency, performance and effectiveness, and this creates misunderstandings as to how productivity is to be measured in agile software processes (Trendowicz & Münch 2009:192). What productivity means also depends on perspective (Petersen 2011:19) and context (Tangen 2005:35). Lastly, there is no consensus on the ideal way to measure software productivity, in both agile and

traditional software development teams. This is due to the fact that software development involves a human being as the creator of software, of which human being activities are not predictable, leading to difficulties in coming up with a reliable definition for software productivity (Trendowicz & Münch 2009:193). Given that software is increasingly the engine that drives enterprises, the differences in concepts prevent any precise way to measure and define software productivity.

Amidst the confusion surrounding the subject of productivity, several other characteristic features were identified by Tangen (2005:39): profitability, productivity, efficiency, effectiveness and performance. The Triple-P model was proposed which aims to distinguish productivity from four other similar terms. The focal point of the Triple-P model is *productivity* (figure 2.2 below) defined as quantity of output (i.e. accurately produced output fulfilling the specifications) divided by quantity of input (i.e. all resources consumed during transformation process). *Profitability* is the business's ability to generate earnings, compared to its expenses and other related costs incurred during a specific period of time. *Performance* is a broad term for excellent manufacturing, which comprises profitability and factors such as speed, flexibility, quality and delivery. In the Triple-P model, *effectiveness* is viewed as the capability of producing the desired outcome and *efficiency* is viewed as minimising waste by making a comparison of what is actually produced, with what can be achieved with the same consumption of resources. Efficiency is an important factor in determination of productivity. However, if high levels of effectiveness and efficiency are both combined during the transformation process, this results in high productivity (Tangen 2002:43).



Ramirez & Nembhard (2004:603) assert that, although the 'Triple-P model' distinguished productivity from other terms such as profitability, performance, effectiveness and efficiency, productivity management in information technology remains a key element due to knowledge activities and is a critical part for making decisions in the world economy (Ross & Ernstberger 2006:31). Productivity metrics in information technology software development are generally based on a quantitative relationship between the delivered size of software and the effort spent to acquire them (Arnold & Pedross 1998:491; MacCormack et al. 2003:79). On the other hand, FP and SLOC are misleading measurements due to the fact that overall development might be unproductive despite the increase in programming productivity (Lee & Schmidt 1997:184). Even so, metrics that are based on code only represent coding activities productivity, whilst non-coding activities, for instance, analysis, design and management are not included in this scope.

In addition, while the agile software development life cycle introduces metrics, viz. Agile Earned Value Metrics (Agile EVMS Metrics) for measuring productivity (Crowder & Fries, 2014:58), there is need for some guidelines on how to use those metrics to measure organisational team productivity. Software productivity as any other agile development practices and concepts - for instance, customer involvement in the development process, incremental and iterative software development, software testing and quality assurance, pair programming, etc. - is surrounded with a lot of controversy of how it should be measured in agile projects, even if the metrics applied in such efforts prove agile to be productive (Mnkandla, 2010:29).

However, developing software is brainwork which involves creating one's knowledge, or using existing knowledge to get the work done (Ramirez and Nembhard 2004:602). Drucker (1994:96) described a Knowledge Worker (KW) as a creative employee who is able to apply analytical and theoretical knowledge, usually obtained through formal learning and work experience, who can create or produce new services and products. Knowledge workers are employees who are working with intangible resources and they should have more knowledge about their job than anybody else in the organisation (Drucker 1999:41-42).

The productivity of a Knowledge Worker is related to *tasks, autonomy, innovation, quality, learning and teaching*, and a knowledge worker is an *asset* (Drucker 1999:41-42). Knowledge worker productivity is not merely looking at the quantity of output produced, quality is also essential. Knowledge workers manage themselves and continuous innovation is part of their work. The research conducted by Ramirez and Nembhard (2004:602) shows that knowledge worker productivity also relates to product quality, customer satisfaction, creativity, profitability, keeping to timelines and team effectiveness and efficiency. Knowledge worker work patterns and productivity were investigated by Palmer (2014:2). He found that knowledge work is required in every workplace and all employees should seek to acquire this type of skills. To that end, educational institutions now focus on lifelong learning to enable learners to acquire the skills required to become productive knowledge workers in this decade (Palmer 2014:3).

Thus, the definition of knowledge worker productivity goes further than the Triple-P model, because of the inclusion of people factors. Knowledge worker productivity includes many

dimensions such as profitability, productivity and performance, and other dimensions such as responsibility, autonomy, creativity and learning have to be added. The new dimensions are therefore a sign of increase in software development productivity of individuals and teams.

Within the context of software development, the definition of productivity is mostly associated with metrics. Many studies adopt the definition based on ratios, instead of productivity theory and its dimensions. The above discussion will help to answer the present research question, viz. which metrics are suitable for determining productivity in agile teams.

2.5.1 Measurement of software productivity

Measurement is generally aimed at control and certainty. Certainty implies comprehending the nature of the phenomenon in order to control, evaluate, or influence a phenomenon (Scacchi 1994:283). The need for measuring software productivity indicates problems encountered by the scientific and systematic process inquiry. The need to measure software production implies a desire to look at fundamental problems, viz. the role of measurement in theory, hypothesis testing, and verification and evaluating performance. It further suggests getting to understand the association between instrumentation and measurement, that is, the artifacts used to gather or measure the data of the phenomenon under study (Hurley 1995:275).

Productivity in much of research outside and inside the software engineering industry is generally expressed as a ratio of produced output units per input effort unit (Hurley 1995:276). Even though there are well established measures of productivity, they vary from organisation to organisation (Chemuturi 2009:13). There are universally accepted measures for effort spent: person-hours, person-days, person-months and person-years, but there are not yet any universally accepted measures for software output (Chemuturi 2009:13).

A sample of studies of software productivity measurement shows that there is often substantial difference with regard to consistency and the use of accepted analytical methods. In the manufacturing industries, productivity is measured for each activity at a time, while the software

industry has not yet come up with universally accepted measures of software productivity (Chemuturi 2009:14).

The problem of measuring productivity often presents clues on how to improve the software development process (Petersen 2011:19; Zelkowitz 2003:197; Trendowicz & Münich 2009:243). The concept of software productivity goes beyond a theoretical abstract. Software productivity is crucial in software engineering, and a better understanding of productivity has become critical in systems analysis considering that good analysis in turn improves software productivity, while software productivity is a success measure of systems analysis (Dilawar 2011:48). Software productivity enables the definition of a baseline for the improvement in software development companies that can then evaluate implemented improvements. Measured software productivity acts as a benchmark in determining whether there is need for improvement to be competitive (Petersen 2011:22).

In his systematic review (Petersen 2011:334-336), defined past and future productivity measurement as *reactive* and *predictive*. The reactive measurement is based on past data to determine productivity. The predictive measurement is concerned with quantifying future productivity. Both reactive and predictive measurements are important in determining whether corrective action is needed, and they help managers in evaluating improvement alternatives (Petersen 2011:340).

As a famous saying from Lord Kelvin states, “*When you can measure what you are speaking about ... you know something about it, but when you cannot measure it ... your knowledge is of a meagre and unsatisfactory kind ...*” Therefore, to acquire a clear understanding of which are the variables that affect software productivity, all stakeholders involved in measuring should be in a position to define software productivity, know who is responsible for measuring and collecting the productivity data, know the parameters that need to be measured, and devise ways to improve productivity (Scacchi 1994:69).

Productivity data in software development should be collected and measured by programmers, team or project managers, automated performance monitors, and observers or outside analysts

(Scacchi 1994:69). To better estimate software productivity, it should be measured throughout the entire life-cycle, not only at completion (Dilawar 2011:54).

2.5.2 Historical software measurement techniques

The oldest metric for measuring software projects productivity is “lines of code” (LOC). The metric was introduced in the 1960s and was utilised for productivity, economic and quality studies (Jones 2008:2). The economics of software applications was measured by using “dollars per LOC”. The measurement of quality was in terms of “defects per KLOC” where “K” (for “kilo”) represented a symbol for “1000 lines of code.” The LOC metric was seen as effective enough to measure the economy, productivity and quality in software applications. Then, the LOC metric analysis was considered useful for all three purposes. The LOC metric stayed the same, as the software industry advanced, but eventually became less useful and even harmful by the year 1980 without anybody realising it. According to Jones (2008:2), “the first known problem with the LOC metric was in 1970 when many IBM publications groups exceeded their budgets for that year.” The line of code use as a key measure in software process remains a controversial issue as debate around its applicability and validity continues. On the one hand, LOC proponents claim that it is an “artifact” of software projects which can be counted easily, that a number of existing estimation software models use KLOC or LOC as main input, and that plenty of data predicated and literature on them already exist (Pressman and Maxim 2015:710)

In contrast, opponents disagree and argue that lines of code as a measure depend on the programming language, and that when productivity is considered, LOC measures penalise well-designed shorter programs; that they are not able to accommodate nonprocedural languages; and that their use in estimation requires a level of detail that may be hard to achieve.(Pressman and Maxim 2015:710). Because counting “lines of code” was seen to be unreliable, other metrics were considered (Schach 2007: 70).

A similar, but independently developed, metric for the size of a product was developed by Albrecht (1979:14) based on *function points* (FP). Albrecht’s metric was calculated as the “number of input items, *Inp*, output items, *Out*, master files, *Maf*, and interfaces, *Inf*” (Schach

2007:256– 257). Function points are a “measure of the product’s size and therefore they can be used for cost estimation and productivity estimation.” Jones (1987:2), stated that, although experiments that were conducted for measuring software productivity rates using function points indicated a better fit than using LOC:, errors in excess of 800 % counting LOC were observed, while only 200 % were observed in counting function points.

The FP, like the LOC measure, remains controversial. The proponents of FP believe that it is programming-language independent, is suitable for applications using nonprocedural and conventional languages, and that its data are most likely to be available earlier in project development, making FP more attractive than LOC as an estimation method. The opponents of function points disagree and argue that it’s just a number with no physical meaning, computation is subjective rather than objective data, and that some dimensions can be difficult to collect after the fact (Pressman & Maxim 2015:711).

2.5.3 Reasons for measuring software productivity

Scacchi (1995:462) suggests that software productivity should be identified and measured in order to reduce cost of development, improve software quality and improve the software development rate.

Scacchi also points out some of the merits of measuring software productivity, as follows:-

- An increased amount of work successfully achieved by the current employee effort.
- Completion of the same amount of work with fewer employees.
- Development of products of superior market value and complexity using the same staff load.
- Avoiding employment of additional staff when trying to increase workload.
- Rationalising capital-to-employee investment levels.
- Reducing bugs in delivered products, and decreasing the amount spent on servicing technical debt of fixing bugs.
- Software production operations should be downsized or streamlined.
- Product defects can be identified in early stages of development.

- Resource utilisation patterns need to be identified in order to discover underutilised resources and production bottlenecks.
- Give awards to responsive or high-output personnel
- Identify underperforming personnel for additional training.

It is impossible to try and accomplish all of the above aims using a single productivity measurement program. The diversity may lead to misunderstandings over how and why software productivity should be measured which might, in turn, result in a situation where the measured results are viewed as suspect, misleading, or inaccurate (Kitchenham *et al.* 2007:10). In this regard, productivity measurement programs should be designed carefully, to avoid misunderstandings (Scacchi 1995:462).

According to Pressman and Maxim (2015:724), “measurement results in cultural change.” To begin a measurement program, the following three steps should be implemented: data collection, metrics analysis and metrics computation. By creating a metrics baseline, software managers and engineers can acquire better understanding of the work they perform, and of the end product they deliver (Pressman & Maxim 2015: 724).

The above discussion will help in this research to determine which agile metrics relate to productivity metrics.

2.5.4 Software measurements and productivity factors

An analysis of the factors related to software development productivity (e.g., measured in effort per SLOC) was first conducted by Walston and Felix (1977:54). The studied factors decreased in significance over the years. For instance, “*chief programmer team usage*” is no longer in common use nowadays. However, some of the factors, such as previous programmer experience with a programming language, user participation, and overall program design constraints, are still valid (Wagner & Ruhe 2008:16).

In 1979, Albrecht (1979:83) proposed “*function points*” as a measurement of software productivity. An analysis of factors like product size and programming language used was conducted. Brooks (1981:3-9) decided to use Walston and Felix’s (1977:54) factors as the base in his IBM study. Brooks (1981:9) discovered that the effects of structured programming and program complexity were important.

An analysis was conducted by Jones (1986:39) which included “numerous productivity factors over numerous domains” and provided industry averages. Jones (1986:39-63) focused his measures intensively on lines of code and function points.

DeMarco and Lister (1987:4) declare that technological problems are not so major compared to sociological problems. Turnover was considered to be among the most important factors that influence productivity, and the importance of an ideal workplace having all the necessities, for example, good lighting system, no noise and others (DeMarco & Lister 1987:17). The programming language used, programmer experience, software defects and remuneration according to DeMarco and Lister (1987), do not bear any significant effect on productivity according to their own view. DeMarco and Lister (1987: 22) went ahead and commented that quality, higher than required by the customer, is an opportunity to higher productivity. The concept of a “jelled team” was introduced, where a “jelled team” was defined as a team that has strong lines of communication and aligned goals (Pressman & Maxim 2015:692). DeMarco and Lister (1987:24) were the first to provide all-inclusive work to date on the soft factors that influence software development productivity.

Boehm *et al.* (2000:15-39) conducted a detailed research project on productivity and the factors that influence it, based on COCOMO II. Their identified technical factors are as follows:

- Similarity of projects
- Software reliability
- Size of the database
- Complexity of the product
- Code reusability
- Documentation in line with life-cycle requirements

- Constraints, i.e. execution time and main storage
- Software tools used
- Platform instability

The various soft factors were analysed by Boehm *et al.* (2000:40-41) who discovered that, when combined, soft factors are more important than the previously mentioned factors. The following were identified as soft factors:

- Team collaboration
- Capability of the analyst
- Capability of the programmer
- Employee turnover
- Experience with different applications
- Experience with using language and tools in use
- Multisite development
- Required development schedule

Maxwell and Forselius (2000:80) claimed that the factors that influence productivity depend on the business domain for which the software was developed. Kitchenham and Mendes (2004:15) established that software reuse has a significant effect on productivity, but the level of reuse is not that essential. Kitchenham and Mendes (2004:16) also suggested that productivity is almost the same in different countries.

In a survey conducted by Berntsson-Svensson and Aarum (2006:3), of factors that influence product and project success, it was discovered that the definition of success differed among industries. Their conclusion, however, was that the recognised influencing factors were the same as in prior research studies: detailed project scope, complete and precise requirements, accurate schedule estimates, involvement of the customer in development, and the hiring of additional personnel.

Ramirez and Nembhard (2004:602) carried out an analysis of the basic category of “knowledge worker” productivity. Knowledge workers (KW) are software developers as opposed to manual

workers. Ramirez and Nembhard (2004:627) point out that there seems to be mutual agreement that there are no practical and effective methods to date that can measure knowledge worker productivity.

Trendowicz and München (2009:185) also conducted research on factors influencing software development productivity. A detailed summary of productivity factors which were recently acknowledged by software industry experts was presented by Trendowicz and München (2009:240-241). Their major result was that the success of software development projects still depended upon the individuals who work on it. Investing in employees is still considered important as it brings more rewards than investing in only methods and tools (Blackburn *et al*, 2000:213-214).

Based on the above literature review, the software productivity factors effect on productivity differs from one researcher to another. Wagner and Ruhe (2008) mentioned previous programmer experience as having an effect on productivity whilst DeMarco and Lister (1987) beg to differ in their own opinion; they think that it bears little significance. Yet, Boehm et al (2000), share the same similar view with Wagner and Ruhe (2008), that capability and experience of the programmer is important when it comes to software productivity. DeMarco and Lister (1987) were the first researchers to present soft factors (e.g. ideal work place) as factors that influence software development productivity; Boehm et al (2000), also introduced similar soft factors that influence productivity. These researchers agreed that soft factors are more important than technical factors when it comes to productivity in software development.

Business domain was also pointed out by Maxwell and Forselius (2000) as an important factor that has effect on productivity, which DeMarco and Lister (1987) found it as a necessity. Kitchenham and Mendes (2004) found out that software reuse had a significant effect on productivity whilst Boehm et al (2000) had a different view as code reusability bears little significance according to their research results. Brentsson-Svensson and Aurum (2006) recognised influencing factors such as hiring additional personnel as having a positive effect on productivity, which Trendowicz and Munich (2009) agreed with them as they found out that software projects success solely depended on individuals who develop it.

The above section gives a comprehensive discussion of the factors found in the literature which influence team productivity. The following section attempts to find out which factors influence agile team productivity and to what extent these factors effect team productivity from the agile team point of view.

2.5.5 Overview of software productivity factors

Understanding the factors that affect software productivity is one the most challenging tasks that face many software development organisations today. The literature has come up with numerous factors that affect productivity, and even with metrics to measure the work products delivered, but productivity levels are still the same. Figure 2.3 below sums up the factors that have profound influence on software productivity. The motivation and skills of people, according to Boehm (1981:13), are the most influential factors in performance, productivity and quality. Product complexity can also have substantial effect on team performance, productivity and quality. Technology and process also have an effect on productivity.

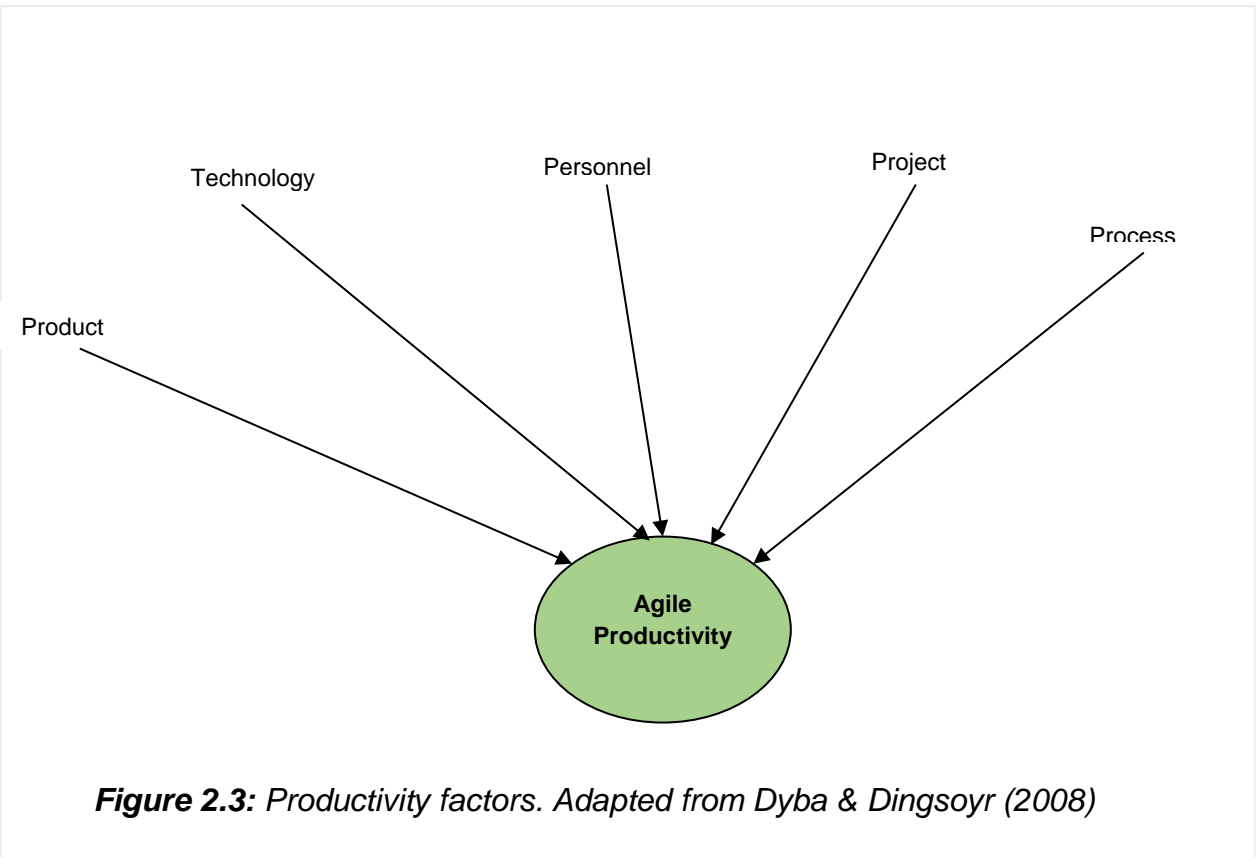


Figure 2.3: Productivity factors. Adapted from Dyba & Dingsoyr (2008)

Trendowicz and München (2009:185-241) reviewed commonly used factors obtained in the published literature that affect productivity. Factors presented first were the same across the whole development context. The most common factors were selected in the context of a certain model, development type and the domain. Table 2.2 shows a summary of the main factors noted from Trendowicz and München (2009) reviews.

Table 2.2: Software productivity factors (Trendowicz and München 2009)

Common software productivity factors		
Cross-context factors	Influence factors	Context factors
	<ul style="list-style-type: none"> ✓ Capabilities and experience of the team ✓ Complexity of the software ✓ Constraints in a project ✓ Usage of tool and quality/effectiveness 	<ul style="list-style-type: none"> ✓ Programming language ✓ Domain ✓ Development type
	Model-specific factors	
	<ul style="list-style-type: none"> ✓ Team capabilities and experience ✓ Tool usage and quality/effectiveness 	<ul style="list-style-type: none"> ✓ Programming language ✓ Domain ✓ Development type

Common software productivity factors		
Content-specific factors	<ul style="list-style-type: none"> ✓ Team size ✓ Project constraints ✓ Process maturity & stability ✓ Methods usage 	<ul style="list-style-type: none"> ✓ Life cycle model ✓ Target Platform
	Development-specific factors	
	<ul style="list-style-type: none"> ✓ Team capabilities and experience ✓ Tool usage and quality/effectiveness ✓ Reuse ✓ Product complexity ✓ Management quality and style ✓ Team motivation and commitment ✓ Required software quality ✓ Tool usage ✓ Method usage 	<ul style="list-style-type: none"> ✓ Programming language used ✓ Target platform ✓ Domain ✓ Development type
	Domain-specific factors	
	<ul style="list-style-type: none"> ✓ Team capabilities and experience ✓ Team size ✓ Reuse ✓ Tools usage and quality ✓ Methods usage ✓ Project constraints ✓ Software complexity 	<ul style="list-style-type: none"> ✓ Programming language ✓ Domain ✓ Development type ✓ Life cycle model
Reuse-specific factors	<ul style="list-style-type: none"> ✓ Quality of reusable assets ✓ Asset complexity ✓ Team capabilities and experience ✓ Product support (from Supplier) ✓ Required software quality 	<ul style="list-style-type: none"> ✓ Life cycle model ✓ Programming language ✓ Domain

The research identified three most literature reviews (Trendowicz and Münich 2009, Boehm *et al* 2000 and Dyba & Dingsoyr 2008) with common factors that have an effect on productivity in software engineering. A summary of the software productivity factors and their effect on productivity from various researchers is presented below in Table 2.3.

Table 2.3 Summary of factors and their effect on software productivity

Productivity Factor	Positive effect	Negative effect	Neutral
Product factors – Development-specific factors			
Code reuse	Trendowicz & Münich, 2009.		
Software characteristics	Boehm <i>et al</i> , 2000.		

Productivity Factor	Positive effect	Negative effect	Neutral
Requirements stability	Maxwell and Forselius, 2000.		
Personnel factors – Domain-specific factors			
Team experience and capabilities	Blackburn <i>et al</i> , 2000; Trendowicz & München, 2009; and Maxwell & Forselius, 2000.		
Motivation	Boehm, 1981 and Boehm, <i>et al</i> 2000		
Project Management	Blackburn <i>et al</i> , 2000 and Boehm <i>et al</i> , 2000		
Programming language			Maxwell <i>et al</i> , 1996
Project factors – Cross-content factors			
Resource constraints		Maxwell <i>et al</i> , 1996	
Schedule		Blackburn <i>et al</i> , 2000; Boehm <i>et al</i> , 2000 and Maxwell <i>et al</i> , 1996.	
Team composition		Blackburn <i>et al</i> , 2000 and Maxwell <i>et al</i> , 1996.	
Process factors – Model specific factors			
Customer participation		Maxwell & Forselius, 2000.	
Daily builds	MacCormack <i>et al</i> , 2003		
Documentation	Boehm, 2000 and Jones 2000.		
Early prototyping	Blackburn <i>et al</i> , 2006 and MacCormack <i>et al</i> , 2001 & 2003.		
Modern programming language	Boehm, 1981		Blackburn <i>et al</i> , 2000
Technology Factors – Reuse-specific factors			
Programming language abstraction			Maxwell <i>et al</i> , 1996 and Blackburn <i>et al</i> , 1996.

The above discussion will assist in finding out which factors have an effect on agile team productivity and to what extent do these factors affect the team productivity level.

2.6 Productivity in agile teams

A team consists of a group of individuals that share a common definition of success, a common goal and culture, and a “sense of eliteness” which makes them unique (DeMarco & Lister 1999:123). According to Avison & Fitzgerald (2006:143), they stated that the team of people and how they work together is perhaps the most important factor in the agile approach. The most effective teams allow diversity by combining a variety of different skills. The agile school recognises that abilities and attitudes of individual programmers and developers are important and “good people provide good outputs and also apply intelligence and are able to act flexibly, which is an important aspect of agile approaches” (Avison & Fitzgerald 2006:144).

The foundation of agile software development is about small teams that work together in a co-located environment, developing crucial software (Abrahamsson 2005:1). Agile software development depends on self-organised teams which consist of highly skilled team members who are innovative and motivated, collaborate through team work, and are self-directed active learners. The agile philosophy emphasises the competency of the individual team member as well as the collaboration of the group as critical success factors for every team. For agile teams to be effective, they make use of the individual team members’ competency and of team collaboration in projects, which makes agile teams self-organising (Pressman & Maxim 2015:692).

A number of agile frameworks such as Scrum and Extreme Programming give an agile team substantial autonomy to make the project-management and technical decisions required, to perform the tasks to completion. Planning is minimised and the team can select their own approach, for example, methods, processes, tools, etc., constrained only by organisational standards and business requirements. During the project activities, a team self-organises to enable it to rely on the competency of each individual team member, in a way that is most valuable to each project at any given time. To achieve this, daily stand-ups meetings are conducted by agile teams to coordinate and synchronise the tasks to be completed on that particular day (Pressman & Maxim 2015:692).

The goal of a team's existence is to work together and perform tasks better than an individual alone. Accordingly, individual member productivity is not as valuable as team productivity. In other words, the evaluation of individual member productivity may not affect knowledge worker productivity (Bosch-Sitjstema et al. 2009: 533). Such assumptions provide a motivation to study team productivity, not individual members. Moreover, evaluating productivity at a team level prevents measurement dysfunctions, since team goals are the core target, not individual work (Melo et al 2011:59).

In conclusion, *knowledge worker productivity* was presented as the best way to describe productivity in the context of software development, including agile teams. In this day and age, productivity goes beyond measuring the ratio between outputs and inputs. Other factors like human factors play a significant role in determining productivity.

2.7 Summary

This literature review chapter presented a discussion of the agile software development paradigm and of the main theoretical concepts regarding agile software development, in contrast with traditional software development. The agility concept was presented based on a literature review. Definitions of software productivity were also presented, and knowledge worker productivity was adopted as a formal definition of productivity of agile team. Some of the most important variables relating to productivity were presented: definitions, monitoring, measurements and factors. The variables identified (i.e. definitions, monitoring, etc.) were discussed in the agile-work context: productivity of teams in dynamic environments, internal processes, for example monitoring and popular metrics.

However, there are research challenges to be noted in agile software productivity evaluation, and the following challenges are the foci of this research in the context of agile team productivity.

The first research challenge is working with a non-established definition of "agility". What is needed is to explore and evaluate current definitions and then verify how the companies under study fit into these definitions.

The second challenge is to review the relationship between productivity and agility. There is a need to balance productivity and agility in short- and long-term perspectives in order to develop strategies to cope with productivity evaluation in agile teams.

The third challenge is to become clear how important productivity is for agile teams. An investigation of the importance of productivity needs to be carried out for companies under study adopting agile methods in South Africa.

The fourth challenge is the measurement of software productivity. Productivity measurement must be designed in such a way as to avoid creating mistrust, conflicts, or other circumstances that create mismeasurements within the software development projects under study.

The fifth challenge is that there are many productivity factors presented in the literature that affect an agile team. Having noted that, a researcher needs to focus her data collection and analysis on the important factors and not to get carried away during the research process. The following strategies will be developed to cope with this issue: (i) a conceptual framework will be created to guide the data collection and analysis; and (ii) two agile teams in different companies and geographical location will be studied to establish the emergence of more influential productivity factors.

The next chapter will discuss the research methodology which is to focus on research approaches and strategies. The chapter will discuss advantages and challenges of approaches adopted for this dissertation e.g. case study, etc.

Chapter 3: Research methodology and design

3.1 Introduction

In the previous chapter, a detailed literature analysis on agile team productivity factors was presented. Chapter three aims to discuss the research approach and methodology to be used in this research study. A detailed discussion of the nature of research and research methodology is provided in this chapter. The chapter also addresses the rationale for the choice of the research strategy and data generation methods chosen for this research.

3.2 Research methodology

Oates (2009:5) defined research as creating new knowledge, by use of appropriate processes, to satisfy the users of the research, and it usually involves analytical thinking that provides results. Research is an original investigation or a creative activity undertaken in order to generate new knowledge and understanding in a particular field of study (Myers 2009:6). According to DePoy and Gitlin (2015:2), research is a systematic way of thinking and acting which is not 'owned' by a single profession or field and can be used and learned by anyone. Research methodologies help researchers by giving them guidelines to minimise inclination and subjectivity in their investigations. Moreover, they provide a connection between the research questions and the data to be gathered so as to answer them (Leedy & Ormrod 2005:6).

This section on research methodology discusses the theoretical framework for qualitative research. The research process used for conducting this research is discussed in detail. The reasons for choosing this particular research method and these data generation techniques are also discussed comprehensively. The research objectives, research questions, research problem, and outcomes will be discussed first. The data collection and analysis process proposed for the research follow. This study is based on qualitative research analysed by an interpretivist method, using the case study as the data collection method (Oates 2009:144).

3.2.1 The research context

According to Oates (2009:6), good academic research should have *sufficient* data sources, *appropriate* data processes and the findings *properly* presented with no hidden assumptions. Therefore, the philosophy underlying the researcher's choice of research question and the way of answering the research question can even depend on individual views and how the individual might investigate the research question.

The following research questions were formulated for this study (see section 1.4):

RQ1. What are the factors influencing the productivity of agile teams and how do these factors affect team productivity from the team point of view?

SubRQ1.1. Which agile practices have an effect on a team's productivity?

RQ2. What is the importance of productivity on companies that adopt agile methods?

SubRQ2.1. How would they define software productivity?

RQ3. What are suitable metrics for determining agile team productivity?

SubRQ3.1. In scenarios where such metrics have negative effects, are there adjustments that could be made to promote productivity?

RQ4. How should productivity factors in agile teams be monitored, considering agility and adaptability?

SubRQ4.1. How do agile metrics relate to productivity metrics?

The research questions addressed above are exploratory (Oates 2009:143) and a case study is to be used to help investigate a real-life instance of software development by teams using agile methodologies. There is not much in the literature about existing research in productivity of agile software teams, although there have been many studies conducted relating to the productivity of software teams.

3.2.2 The potential research outcome

In order to establish the research outcomes, an analysis of the research objectives was conducted. The research objectives in Chapter 1 were identified as follows:

Main objective

- To develop a framework that gives a better understanding of the factors that influence the productivity of agile teams, the importance of productivity in agile methods and ways in which the agile metrics can support productivity metrics.

The secondary objectives were identified as follows:

Table 3.1 Secondary objectives

Secondary objective	Methods of implementation
1) To define software productivity and finding out its importance in companies adopting agile methods	<ul style="list-style-type: none">• Document analysis through focused literature search.• Semi-structured interviews with identified companies using agile methods in South Africa.
2) To determine factors that influence agile team productivity and their actual influence.	<ul style="list-style-type: none">• Reviews of related literature• Observation of agile teams• Reviews of publications• Semi-structured interviews with identified companies using agile methods in South Africa.
3) To determine which metrics are suitable for measuring productivity in agile teams	<ul style="list-style-type: none">• Reviews of publications• Reviews of related literature• Semi-structured interviews with identified companies using agile methods in South Africa.
4) To find out ways of monitoring productivity factors in agile teams, considering agility and adaptability	<ul style="list-style-type: none">• Reviews of related literature• Reviews of publications

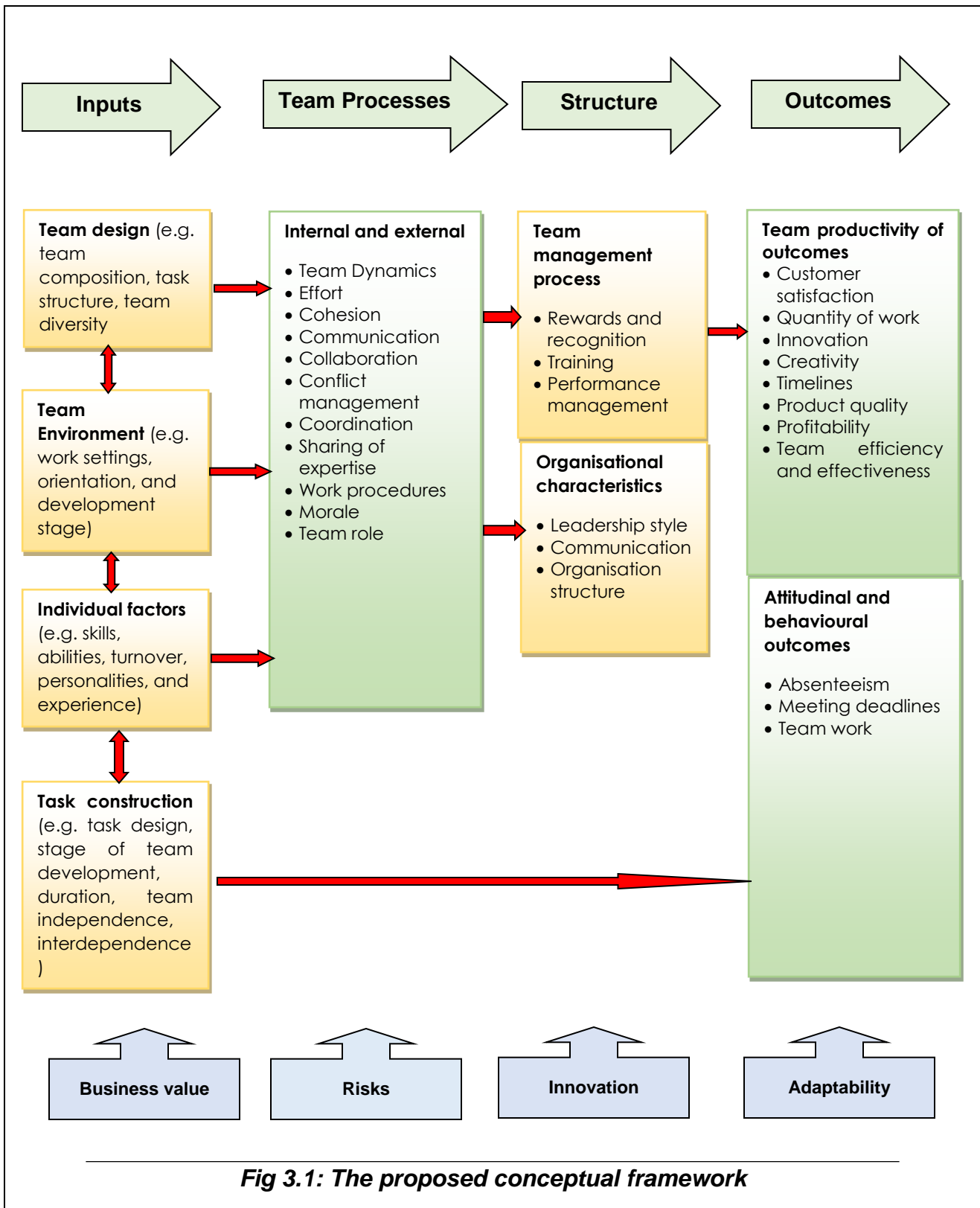
Secondary objective	Methods of implementation
	<ul style="list-style-type: none"> • Semi-structured interviews with identified companies using agile methods in South Africa.
<p>6) To explore agility properties to better understand how to define and evaluate software productivity in a highly flexible, adaptive and response environment.</p>	<ul style="list-style-type: none"> • Reviews of related literature. • Reviews of publications. • Semi-structured interviews with identified companies using agile methods in South Africa.

The theoretical framework and the research model process below will assist in coming up with the prospective or potential research outcomes.

3.2.3 Theoretical framework or proposed conceptual framework

A conceptual framework was defined by Miles and Huberman (1994:18) as a written or visual product that explains in a narrative or graphical form the things to be studied, such as key concepts, factors, or variables, and the assumed relationships between them. A conceptual framework allows researchers to make reasoned, defensible choices about how we might explore topics or themes heretofore underexplored or to explore old questions in new contexts (Ravitch et al. 2012:13). It matches our research question with those choices, and in turn aligns our analytic tools & methods with our questions. This study investigates factors that influence productivity in agile teams and how productivity is currently being measured.

Using the factors, characteristics and processes of software productivity in agile teams, the resulting proposed conceptual framework is depicted in figure 3.1 below.



The above conceptual framework to investigate the factors that influence productivity in agile teams and how productivity is currently measured, is useful to provide clarity and focus in this research study as it drives further discussions around the results (Miles & Huberman 1994). The conceptual all aspects discussed in the literature review, providing details on factors influencing software productivity of agile teams.

Input factors: are critical success factors identified in literature as an important aspect of agile software development. Input factors comprises of team design, team environment, individual factors and task construction. The most relevant team member characteristics are knowledge, skills, motivation and personality, while team characteristics include size, diversity, staff turnover, and shared beliefs. Team capabilities and skills are the most significant personnel characteristics that influence software productivity (Maxwell & Forselius 2000) and usually play a moderator role in frameworks that aim to explain productivity variations (Blackburn *et al* 2000, Trendowicz & Münich 2009). Recruiting the right personnel with appropriate skills and empowering them are critical for agile development success (Highsmith 2004).

Team design: entails how the team is structured which includes the number of team members per team (e.g. 4 to 7 members), how the each task is structured, and the level of seniority in each team (e.g. junior to senior programmers) as per a given project.

Team environment: entails the team working space; which includes the work settings (open office, sitting arrangements, etc.), orientation (location e.g. client site) and their software development stage (e.g. initial phase or final phase).

Individual factors: includes programmer skills and abilities to successfully complete assigned tasks within a team. It also encompasses the staff turnover (low or high), personalities of each individual team member and the level of developers' experience within the team.

Task construction: it involves task design, stage of team development, duration, team independence and team interdependence.

Team processes factors: interactions among team members, management, customers and suppliers directly affect team performance. Examples of processes are team dynamics, effort, cohesion, communication, team collaboration, conflict management, coordination, sharing of

expertise, work procedures and team morale. Since agile focus on people, teamwork, and their interactions through agile practices, the mentioned processes might have significant influence on team productivity and therefore were included in the conceptual framework.

Structure: team management process and organisational characteristics are important factors to consider in agile team productivity. Reward and recognition, training and performance management play a critical role in productivity of agile teams.

Outcomes: there are some expected outcomes, including customer satisfaction, product quality, agile team productivity and attitudinal and behavioural indicators. As noted from the literature review (Chapter 2, section 2.5.1), that productivity is difficult to measure, the researcher consider agile team productivity from the team perception of their overall team productivity. The attitudinal and behaviour outcomes were considered because of their importance in establishing agile self-organisation and teamwork (Moe *et al.* 2008, 2010).

3.2.4 The research approach

Empirical studies may apply different research methods. Research is broadly categorised into two main classes, that is, fundamental (basic) and applied research (Rajasekar *et al.* 2013:7). The fundamental or applied research can either be quantitative or qualitative or even both. Quantitative research is commonly associated with measurement of quantity or amount, whilst qualitative research usually uses a phenomenon involving quality. Quantitative research is basically “used and analysed by positivist researchers” (Oates 2009:255), whilst qualitative research is primarily “used and analysed by interpretive and critical researchers” (Oates 2009:266). Quantitative research methods use a deductive approach to data analysis whereas qualitative research methodologies use an inductive approach. Deductive reasoning is a top-down approach where reasoning works from more general to more specific hypotheses that can be tested. Inductive reasoning is a “bottom-up” approach where the researcher moves from specific observations to general conclusions and theories. The two approaches above indicate that the nature of this research study is inductive. Inductive and deductive reasoning methods have a completely different “feel” to them when carrying out research. Deductive reasoning is

narrow and is concerned with confirming or testing hypotheses whilst inductive reasoning is more exploratory and open-ended (Trochim 2006:17).

Table 3.2 summarises the main characteristics of quantitative and qualitative research methods.

Table 3.2 Characteristics of quantitative and qualitative research (Rajasekar et al. 2013:9)

Quantitative Research	Qualitative Research
- <i>It is numerical, applies statistics or mathematics, is non-descriptive and uses numbers.</i>	- <i>It is non-numerical, applies reasoning, is descriptive and uses words.</i>
- <i>Evidence is evaluated because it is an iterative process.</i>	- <i>Its aim is to get feeling, meaning and describe a situation.</i>
- <i>The presentation of the results often takes place in graphs or tables.</i>	- <i>Data in qualitative research cannot be graphed.</i>
- <i>Quantitative research is conclusive.</i>	- <i>Qualitative research is exploratory.</i>
- <i>Quantitative research investigates the where, what and when of decision making.</i>	- <i>Qualitative research investigates the how and why of decision making.</i>

Table 3.2 above shows various characteristics of each research method. Quantitative research can be categorised into experimental and non-experimental research (such as surveys), and its basic building blocks are variables. The purpose of experimental research is to study cause-and-effect relationships, while non-experimental research does not involve manipulation of independent variables nor random assignment of group participants. For this study, quantitative research was not considered since it uses postpositivist claims of knowledge development, uses research strategies such as surveys and experiments, and collects numeric data that produce statistical data as findings (Creswell 2003:18). The research under study involves humans as the centres of productivity.

This research study is associated with a qualitative research methodology that is appropriate to answer the questions focused on “what” and “how” (Yin 2008:2), and aims to acquire a good understanding of the problem being investigated. This study has a perfect match with the characteristics of qualitative research

It is intended to explore which productivity concepts agile teams use in their projects; this is predominantly qualitative in nature since it involves non-numeric data and uses words (Oates 2009:266). It is also planned to ascertain which factors have an effect on productivity of agile teams, and to investigate whether these agile practices affect productivity from the teams' point of view. Qualitative research can be used in academic natural sciences and social sciences, as it allows the researcher access to large amounts of “hard data”. Denzin & Lincoln (2005:2) described qualitative research as a “multi-method in focus, comprising an interpretive, naturalistic approach to its subject matter.” Qualitative research is usually more flexible, allowing more naturalness and allows more collaboration and interaction between the study participant(s) and the researcher (Denzin & Lincoln 2005:2). The productivity of an agile team was to be studied in a natural setting (an office where the software development is taking place) to allow the researcher to interpret or make sense of the phenomenon under study, and in turn allows the researcher to answer the research questions for this study with more meaningful information.

3.2.5 The research philosophy

The philosophy of research is a belief about how to gather, analyse and use data about a phenomenon. Holden and Lynch (2004:15) state that reviewing the philosophy is a critical aspect of the research process because it opens the researcher's minds to new possibilities, which normally lead to the enrichment of research skills and increase the researcher's confidence in the selected research strategy. The term ontology (researcher's viewpoint of the nature of reality) as opposed to epistemology (researcher's view about what counts as legitimate, valid and acceptable knowledge) constitutes the various philosophical paradigms of the research approach (Lee 2004:5). There are a variety of research strategies with distinct underlying philosophical paradigms that have been used in information systems, and researchers need to be aware of these paradigms (Oates 2009:282). There are three major divisions of research-

philosophical paradigms, namely positivism (sometimes called ‘the scientific method’), interpretivism, and critical research (Oates 2009:283).

The positivist philosophical paradigm underlies the scientific method which mainly uses experiments, but in information systems, for instance, positivist researchers use surveys as opposed to experiments, as the latter are often not feasible (Oates 2009:286). Interpretivism research is associated with understanding the social context of an information system: the social processes by which it is developed and construed by people and through which it influences, and is influenced by, its social setting (Oates 2009:292). Critical research is associated with identifying conflicts, contradictions and power relations, and empowering people to eliminate them as sources of dominance and alienation (Oates 2009:296).

In pursuance of identifying the most appropriate philosophical paradigm for this research, the three philosophical paradigms: positivism, interpretivism, and critical research, are compared and constructed below.

Table 3.3 Comparison of the three research philosophies (Bristow & Saunders 2015:136)

	Research Paradigms		
	<i>Positivism</i>	<i>Critical Research</i>	<i>Interpretivism</i>
Ontology <i>(nature of being or reality)</i>	The nature of the world is real and exists independently of humans. A social and physical world exists, to be captured, studied and measured.	Layered/ Stratified ontology (the empirical, actual and real) Reality is external and independent. Objective structures are underlying structures that shape the observable events. Causal mechanisms.	Dynamic, rich, socially constructed meaning through language and culture. Multiple subjective meanings, realities, interpretations. Flux of processes, practices, experiences.
Epistemology	Positivism uses scientific method:	The critical researcher embraces	Knowledge is gathered through

	Research Paradigms		
	<i>Positivism</i>	<i>Critical Research</i>	<i>Interpretivism</i>
<i>(what constitutes acceptable knowledge)</i>	Observations and measurements of facts, and produce hypotheses, theories.	epistemological relativism. Knowledge is historically situated and transient. Facts are social constructions that do not exist independently but rather are agreed on by people. Critical researcher focuses on historical casual explanation as contribution.	multiple subjective realities. It focuses on narratives, stories, interpretations and perceptions. Research is aimed at new understandings of people in natural social setting.
Axiology <i>(role of value)</i>	Value-free research. The researcher is neutral, objective and independent, an impartial researcher.	Value-laden research. A critical researcher acknowledges bias by the world views, cultural experience and upbringing. The researcher tries to minimise errors and bias. The critical researcher is as objective as possible.	Value-bound research. Researchers bring own subjective experience and are part of what is researched. Researcher's interpretation is key contribution. Researchers are reflexive.
Methodology <i>(Typical methods)</i>	Deductive, large samples, highly structured, measurement,	Retroductive, in-depth historically situated analysis of emerging agency and pre-existing	Usually inductive Small samples, comprehensive investigations,

	Research Paradigms		
	<i>Positivism</i>	<i>Critical Research</i>	<i>Interpretivism</i>
	typically quantitative data analysis.	structures. Critical researchers use a variety of methods and data types to fit the subject matter.	qualitative data analysis.

Table 3.3 above presents a detailed comparison of the three philosophical paradigms. It is clear, from the table, that the positivism paradigm is more relevant to studying natural world aspects, for example, pressure, atoms, etc., and less commonly suited for researching the social world, which is the world of people, organisation, etc. (Oates 2009:288). The aim of this study is to test theories that are generalizable, using reductionism and repetition. The positivist approach was not considered for this research study as it is not always suitable for studying the social world. This research is not carried out in an objective manner but rather in a subjective manner. The goal of critical research is to concentrate on the conflicts, power relations and contradictions in the current world, and to help eliminate them as causes of dominance and alienation (Oates 2009:297). According to Oates, critical research aims at identifying and challenging conditions of domination, and the limitations and injustices of the status quo and of assumptions taken for granted, which makes critical research unsuitable for this study. As this study aims to identify, explain and explore how factors in a particular social setting (an agile software development team) are related and interdependent, this study adopted an interpretivism approach. According to Checkland and Holwell (1998:22), the researcher then observes people’s perceptions of their world (as individuals or as a community) and tries to get an understanding of phenomena through the meanings and values assigned to them by people.

The choice of the research philosophy for this study raises important methodological implications regarding the choice between various research strategies and data generation methods. This research study involves group and individual software development team members and their productivity; it therefore lies within the interpretivism paradigm as illustrated below:

- Multiple subjective realities: how each software development team member considers how productivity might differ from one company to another. The shared view of individuals, of what constitutes productivity, may differ from one software developer to another.
- Socially and dynamically constructed meaning: for this study, knowledge to be gathered through shared meanings and understanding of the group or individual participants.
- Researcher's reflexivity: the researcher's own values, beliefs, assumptions and actions shaped the research process since the researcher not only gathered data from observing the phenomena, but also from renegotiation of understandings, meanings and practices.
- This study involved studying humans in their natural settings. The researcher aimed to understand how software development teams work and produce results in their natural settings. What happens during software development was studied from the perspectives of the agile software development team.
- Multiple interpretations: in this study, it is clear that more than one explanation or definition of what constitutes productivity will be reached, and the researcher will discuss which one encompasses all the variables.

This research study is highly qualitative, that is, it is essentially interpretive research (Creswell 2003:13). Interpretive research does not validate or invalidate any hypothesis, but instead is concerned with understanding the social context of an information system (Oates 2009:292). This research aims to identify, explore and explain interpretations that participants have with regard to factors that affect software productivity in companies that use agile methodologies. The data collection also provides an account of what happens in a social setting which permits of multiple interpretations that can be created by different participants through interactions in the research process. By examining and interacting with participants, the researcher may develop a deeper understanding of the factors that affect agile team productivity, and how to develop a framework that will assist agile teams in measuring their own software productivity.

3.2.6 Discussion and rationale for the choice of research strategy

There are numerous research strategies that relate to the three philosophical paradigms discussed above. As noted above, this study falls under the interpretivism approach, and there are a few research strategies identified for this research. Experiments and surveys are strongly associated with the positivist paradigm, therefore are not considered for this study. Critical research, at the other end, is concerned with changes in a social situation or individuals and mostly uses design and creation and action research as research strategies, and therefore is also not considered for this research study. The philosophical paradigm of interpretivism is the most suitable approach for this study, as it emphasises a subjective approach to studying social phenomena and uses research strategies such as ethnographies and case studies which are strongly associated with an interpretive paradigm.

There are five major types of qualitative research strategies, as follows:

- (i) *Ethnography*, where a researcher gathers and records data, primarily observational data about the cultural group being studied in a natural setting over a long period of time (Creswell 2003:14). The outcomes of an ethnography are heavily “dependent on the researcher as the research instrument, and critics point out that the researcher is not detached and objective, and the findings are not likely to be repeated by someone else” (Oates 2009:174). Although ethnography is strongly associated with the interpretive approach, for this study, ethnography was not chosen as a research strategy to be used as the research was not focused on studying peoples and cultures.

- (ii) *Grounded Theory*, where a researcher intends to do field research, analyse the data to establish the theory that emerges, so that the theory is *grounded* in the field data (Oates 2009:274). A grounded-theory process is an “inductive approach” as opposed to the “deductive approach”, where a researcher develops an abstract theory first and then goes into the field to evaluate it (Oates 2009:274). For this research study, grounded theory was not selected as a strategy to be used, because it is concerned with generating theories. This research study is concerned with generating data from

a sample of identified companies that use agile methodologies in their software development, hence grounded theory is not applicable it.

- (iii) *Phenomenological research*, where a researcher identifies the core of human experiences regarding a phenomenon, according to the descriptions of participants in a study (Creswell 2003:15). Understanding people's lived experiences makes phenomenology a philosophy as well as a method, and the procedure is associated with studying a small number of subjects through broad and prolonged engagement to develop meaning for relationships and patterns (Creswell 2003:15). Phenomenological research was not selected as a strategy to be used for this study because it is mainly suited for studying emotional, affective, and often intense human experiences (Merriam 2009:26).
- (iv) *Narrative Research* is a research inquiry where the researcher gathers human and personal dimensions of experience over time, and records the relationships between cultural context and individual experiences (Clandinin & Connelly 2000:20). The information gathered is then retold into a narrative chronology by the researcher. Views of life of participants as well as those of the researcher's life are combined together at the end to form a collaborative narrative. Narrative research was not chosen as a research strategy for this study because it focuses on human lives and their stories. Narrative research lacks analysis and trustworthiness as it leans heavily on narration as a form of representation.
- (v) *Case Studies*, where a researcher investigates or explores a contemporary phenomenon within its real-life context (Yin 2003:13). Case studies are bounded by time and activity and the researcher can use a variety of data sources and methods to gather detailed data over a given period of time (Oates 2009:142). In a case study, a researcher obtains detailed information about a particular instance of the phenomenon under investigation (Oates 2009:142). Research conducted using case study is conducted in a natural setting. The research setting is already in existence prior to the researcher conducting the investigation and continues to be in existence

after the completion on the research. A case study approach was chosen as the best and most appropriate strategy for the present research because the research on agile team productivity would then be conducted in a natural setting (the offices where programmers, analysts, etc. do their work each and every day) and also because detailed information needed to be obtained in order to answer the research questions for this study. A discussion of the rationale for choosing a case study as a research strategy for this research is presented in the next section.

3.2.6.1 Applicability of the case study method to this research

A case study concentrates on a specific occurrence of an entity which is to be investigated: an association, a data framework, an examination gathering, a framework designer and so on (Oates 2009:141). The instance, or case, is studied thoroughly, by means of a variety of sources and methods, for example, observation, interviewing, questionnaires and document analysis. The aim is to acquire a thorough insight into the 'life' of that entity and all factors and their interrelationships.

Gillham (2000:1) states that a case study is an investigation aimed at answering specific research questions that aim at diverse types of evidence from the case settings. Yin (2008:18) points out that a case study is an empirical inquiry used to investigate a contemporary phenomenon in its real-life setting, mostly if the boundaries between context and phenomenon are not well defined. According to McMillan and Schumacher (2001:401), a case study examines a case, or limited system, in depth over a certain time, utilising numerous sources of data available in the setting.

A case study can follow an essential philosophical paradigm of critical research, interpretivism, or positivism (Oates 2009:144). The case study is suitable for both theory constructing and testing and therefore the interpretive approach will be used in this research. The case study methodology was chosen as the most suitable approach because it produces data close to the individual's experiences (an agile team is made up of individuals) and it allows the researcher to answer research questions on "what" and "how" (Yin 2008:2). A case study, according to Yin

(2008:18), has the capacity to deal with technically distinct situations where there are a number of variables of interest, as against single data points. Any one result normally relies on several sources of evidence, and data need to converge in a “triangulating” fashion. Some results benefit from previously developed theoretical propositions, to guide data generation and analysis. In the present study, there are many variables involved in the measurement of software productivity as well as factors that influence the productivity of each software development team. The case study research method offers the approach to design logic, data collection and data analysis which makes it the most appropriate qualitative research strategy to be used for this study.

3.3 Data collection and analysis

This research is based on qualitative research which relies on observation, interviews, audio visual and documents for its most common data-gathering methods.(Creswell 2009:173). Interviews and focus group interviews were used for data collection, and a comprehensive discussion of interviews is presented below. Two techniques were chosen so as to enable the researcher to cross-check data from two different angles to help in providing a multi-dimensional view of the collected data.

The following section discuss the chosen data collection methods, viz. interviews and group interviews (of focus groups).

3.3.1 Interviews

According to Yin (2003), interviews are discussions guided by a prepared procedure and are believed to be one of the most vital resources for data collection. Interviews can be categorised into three types: *structured*, where the interviewee has to answer standardised questions; *semi-structured*, where the interviewee has a list of themes, but is allowed to change during the course of the interview to follow up specific themes; and *unstructured*, where the researcher has less control of the interview. In the present case, the researcher will be using semi-structured interviews, to allow participants to bring up relevant issues that might not be covered in the researcher’s prepared questions. Unstructured interviews were not considered because of their lack of structure and order in the interviewees’ way of

answering questions (Davis et al. 2006). The interview questions would focus on the participants' experience of working with agile methods and on their particular roles in agile projects.

Table 3.4 Advantages and disadvantages of interviews (Oates 2009:198-199)

Advantages	Disadvantages
Topics are dealt with in depth and in detail.	Interviewing, transcribing and analysis of unstructured data is time-consuming for the researcher.
Relatively little equipment is needed, and the interviews are built on the researcher's present social skills.	Lack of reliability due to the effect and context of the researcher which makes it difficult to achieve consistency and objectivity.
It gives the researcher an opportunity to cross-check the interviewee to see if he or she is the correct person to be answering the questions.	Can be misleading since they focus on the participant's answer rather than what might really be the case.
Permits flexibility – as the interview progresses, it allows the researcher to adjust the line of inquiry.	Interviewees can give a false impression since they are aware that they are speaking for the record.
Talking to a non-critical listener makes interviewees enjoy the opportunity to speak about their ideas.	They can be upsetting and stressful for both the researcher and the interviewee since they require tact and social skills.
Some participants often prefer interviews as opposed to completing questionnaires, since they can meet the researcher and they find it easier to talk than to write down responses.	They are not generally suitable for research that requires generalisations about a whole population.

3.3.2 Group interviews (focus groups)

The focus group interview is an effective approach to gathering qualitative data. A group of participants and the researcher sit together and discuss certain items depending on the needs of the case, allowing the group members to talk to each other and carry on discussions. From this, new issues might arise which had not been recognised by individual group members (Oates 2009:194-195). The researcher leads the discussions throughout the session, but yields to participants when necessary to engage more in the topic of discussion. This method is efficient since it helps to collect data from several perspectives simultaneously, although it will be challenging to find an occasion when all participants (especially industry practitioners) are available together. Since it is difficult to facilitate a group discussion and keep notes at the same time, audio recordings are used, and the discussion is transcribed.

Table 3.5 Advantages and disadvantages of group interviews (Oates 2009:195)

Advantages	Disadvantages
Consensus views can be generated.	Some team members might dominate the interview and the quiet members may struggle to be heard.
More responses can be generated, as one participant’s ideas and views can be challenged by other participants and stimulate other group participants to new ideas.	Some participants might be reluctant to express their ideas and views in front of others.
It allows the group to brainstorm themes.	Opinions “expressed might be only those deemed to be ‘acceptable’ within the group.”

This research study involves data collection from different participants using agile development methodologies in their software development. Qualitative data analysis is used. In this research, with contributions from two organisations, based in South Africa but located in different provinces.

3.3.3 Data generation methods for this study in context

According to Creswell (2009:4), qualitative research is an approach to understanding and exploring the opinions of individuals and groups affected by a human or social problem. Qualitative researchers normally use open-ended questions, which allows the participants to air their views (Crotty 1998:10). In qualitative research, the researcher seeks to understand the context and setting of the participants by going to that setting and personally gathering information. According to Kumar (2005:123), although there exist several data generation methods in qualitative research, interviews are the most commonly used instrument. For the exploration of the central topic in this research, a semi-structured interview design with open-ended questions was deemed the most appropriate.

The open-ended type of interview was chosen because it gives respondents enough time and scope to express their different views and allows the researcher to follow up and react to unfolding events and emerging ideas. The results obtained through open-ended questions can be compared among each other since each participant is required to express his/her views about the theme under study, which perhaps will not be possible in a survey, for instance. Open-ended questions also permit the respondents to freely voice their experiences, thereby minimising the influence of the researchers' "attitude and previous findings" (Creswell 2005:18).

Both focus group and individual interviews were viewed as the most appropriate data collection techniques for this study because productivity in software development involves individual workers and workers working in a group (team). Having face-to-face interviews with each programmer, system analyst, etc., and group interviews with the whole programming team will help the researcher to ascertain which factors highly affect the team productivity. This will assist the researcher in answering the research questions of chapter 1.

3.3.4 Analysing textual data

Data analysis starts with the researcher reading through the collected data to get the general impression. The researcher then determines key themes in the collected data (Oates 2009:268-269). In the present case, three themes were identified, as follows:

1. Segments that do not relate to the research questions and therefore are not needed.
2. Segments that give general overview information which might be needed in describing the research context (for example, company history).
3. Segments that are directly significant to the research questions.

The research will focus on the last segment, with the researcher categorising each segment or unit of data into themes. The categories will come from existing theories from the literature, or are developed by the researcher (Oates 2009). Computer-aided qualitative analysis software tools are used for tasks such as searching the text, data coding, data grouping, writing tools, etc.

3.3.5 Non-textual qualitative data analysis

According to Oates (2009:273), non-textual qualitative data comprise videos, sound clips and audio tapes multimedia documents and photographs. Raw data from interviews, that is, audio tapes, are indexed and transcribed. For analysing non-textual data, computer-aided qualitative analysis software tools are used for tasks such as transcript creation, coding, data organisation and hyperlink creation.

Qualitative data analysis was chosen for the following reasons:

- The 'slice of data' can be examined and analysed because of its richness and thoroughness, which includes words, images, different websites and expressions from the participants, all of which were relevant to this research with its focus on human productivity, unlike a study that can be expressed as numbers only.

- The possibility exists of more than one explanation, rather than assuming that there can only be one 'correct' explanation (Oates 2009:277). Therefore, the possibility exists of different researchers reaching different valid conclusions from the same research results.

3.4 Research design

Research design refers to a 'master plan' used for planning, collecting and analysing data, which gives direction on how this study is to be carried out. Schumacher and McMillan (1993:31) describe research design as the structure and plan of investigation used to acquire evidence in order to answer research questions. Mouton (1996:175) states that the research design serves to structure, plan, and execute the research and thus to maximise the validity of the findings. In addition, Yin (2003:19) asserts that literally a research design is a plan of action for moving from '*here*' to '*there*', where '*here*' is considered as the original set of questions to be answered and '*there*' is some set of (conclusive) answers.

3.4.1 A Model of the research process

Figure 3.2 summarises the research phases, the methods used and the purposes.

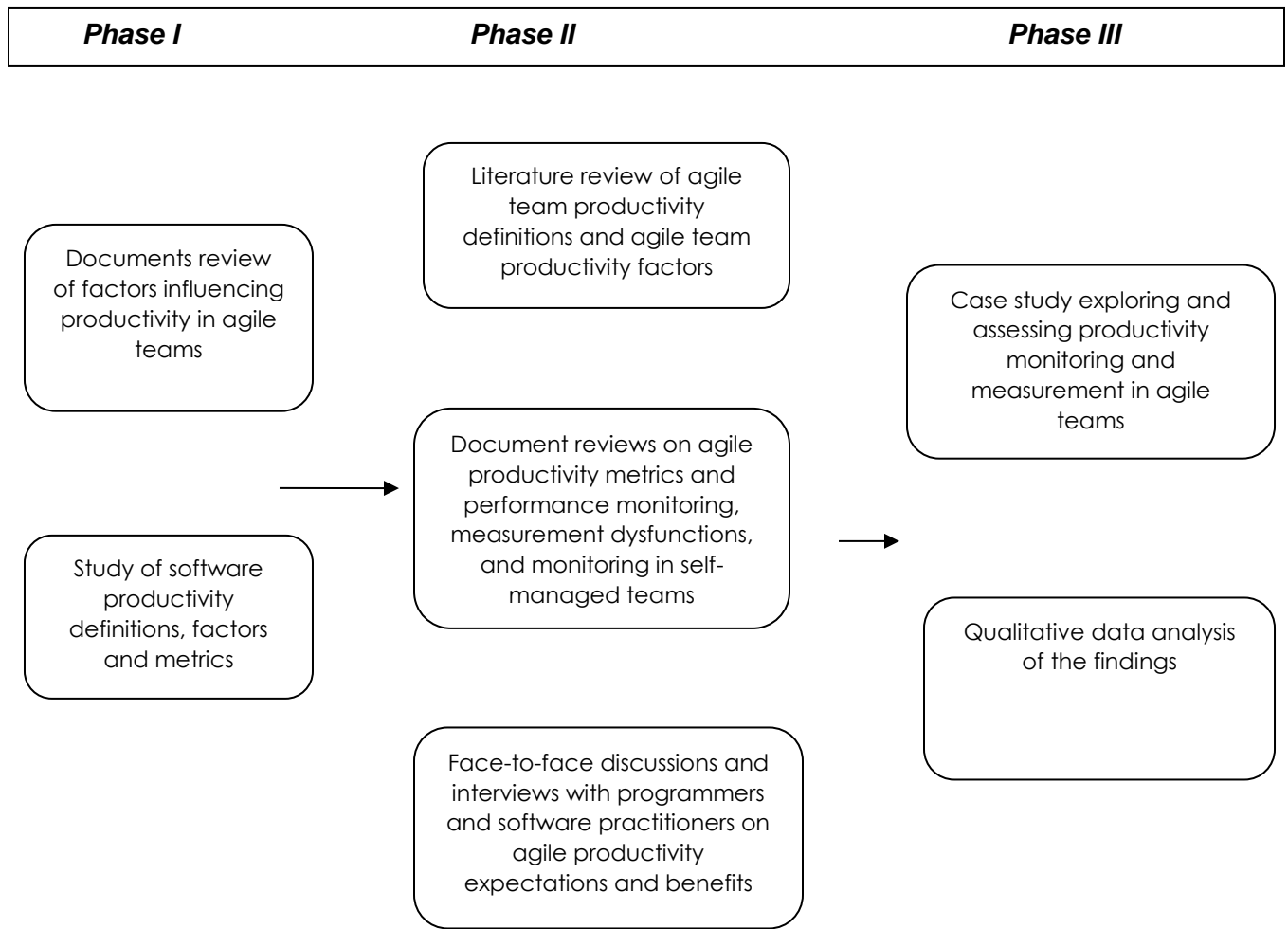


Figure 3.2: Research phases

Firstly, *Phase I* is a combination of a literature review of software productivity and of considered studies. Traditional software productivity analysis, such as recommended by ISO/IEC 15393, can generate misleading analysis (Kitchenham *et al.* 2007:316). Productivity definitions are based on traditional software development approaches, raising questions on the validity of the metrics adopted by companies to evaluate agile projects. In parallel, the researcher will conduct a literature review based on primary and secondary studies of software productivity definitions, factors, and metrics, both for traditional and agile approaches.

Phase II: A case study will be used by the researcher to identify definitions of productivity and the factors that influence productivity in agile teams. Case studies are to be developed to gather evidence on the importance of productivity in agile teams. Thereafter, the researcher will collect qualitative data on how companies are evaluating the productivity of agile teams', and whether they are using metrics for that. The researcher's findings should answer RQ1, RQ2, and RQ3.

Finally, *Phase III* involves monitoring processes for agile team productivity measurements, and the evaluation of their effectiveness. Since organisations that use agile methodologies are still trying to find good ways to monitor their productivity, a case study will be used. The main goal is not to develop new metrics for measuring productivity of agile teams, but to iteratively adopt some proposed agile metrics and evaluate them in a real setting. The researcher's finding should then answer RQ4.

3.4.2 Summary of the research process

Figure 3.2 shows a summary of the research process and the phases involved in this study. The use of a variety of data collection methods enables the researcher to look at productivity in agile teams in different ways. Details of each research process stage are discussed as follows:

Table 3.6 Research stages

Stage	Description	Reference
1	Factors influencing productivity in agile teams.	Chapter 2 2.5 & Table 2.2
2	Software productivity definitions, factors and metrics.	Chapter 2 2.5 & 2.5.5
3	Agile team productivity definitions and agile team productivity factors.	Chapter 2 2.5 & Table 2.2

Stage	Description	Reference
4	Agile productivity metrics and performance monitoring, measurement dysfunctions, and monitoring in self-managed teams.	Chapter 2 2.5.1; 2.5.2; 2.5.3 & 2.5.4
5	Preparing interview questions and finalising them.	
6	Conduction of face-to-face interviews with programmers, software practitioners and focus groups on agile team productivity expectations and benefits. (based on two selected companies under case study in South Africa)	
7	Qualitative data analysis of the findings.	

Stages 1 to 4 were conducted as preliminary work necessary in developing the research questions. Stages 5 to 7 address the research questions. The stages 1 to 7 are summarised below to give an overview of the context of the design.

3.4.2.1 Factors influencing productivity in agile teams

Stage 1 identified and analysed common factors that influence software productivity in agile teams. Table 2.2 shows an overview of the factors that were identified in the literature and which have a positive or negative effect on the productivity of agile teams. Cross-content factors, content-specific factors and reuse-specific factors were identified as the most common published factors in the literature that affect productivity in software development teams.

3.4.2.2 Software productivity definitions, factors and metrics

Stage 2 was to study common productivity-related definitions, factors and metrics used by agile teams. Chapter 2 (section 2.5) presented a detailed discussion of definitions of software productivity which relate to the ratio of produced software divided by the labour and cost of

producing the software. This definition did not exhaust everything that is involved in productivity and therefore the 'Triple-P model' (figure 2.2) was proposed which aims to distinguish productivity from profitability, efficiency, effectiveness and performance. Although the 'Triple-P model' distinguished productivity from the above-mentioned factors, productivity management in information technology remained a key issue due to productivity metrics being generally based on ratios and knowledge worker productivity including people factors. The purpose of this phase or stage was to help answer the research questions based on productivity definitions, factors and metrics used in agile software development.

The software productivity definitions, factors and metrics presented were used as the basis for data gathering. In reviewing each definition, factor and metric, the focus was directed on those definitions, factors and metrics that specifically relate to agile software development.

3.4.2.3 Agile team productivity definitions and agile team productivity factors

Stage 3 identified the productivity definitions and the agile-team productivity factors that formed the research questions. Most of the authors who studied 'productivity' were considered for this dissertation; Jones (1996); Mills (1983); Tangen (2002); Trendowicz & Munich (2009); Petersen (2011); Boehm (2000); Ramirez & Nembhard (2004) were identified as important.

A number of productivity factors were identified by the researcher across all the software development teams, and only those factors that affect agile teams were chosen. Understanding the factors that affect software productivity in agile teams is one of the most difficult tasks that face many software development organisations adopting agile methodologies today. Chapter 2 provides a detailed discussion of these factors and their effect on agile team productivity.

3.4.2.4 Agile team productivity metrics and performance monitoring

In many studies outside and inside the software engineering world, productivity is expressed as the ratio of units of output produced to the units of input effort spent. Although there have been well-established metrics to measure productivity, it still varies from one organisation to another. Two organisations in South Africa, located in two different geographical areas, were chosen for the present research in order to establish which productivity metrics are being used by these organisations and which performance monitoring techniques are in place. The organisations considered for this study have been working with agile development projects for more than three years. The software development manager for each organisation was contacted before setting up interview dates with the entire software development teams. The background of the research and some preliminary interview questions were sent to the software development manager beforehand. The productivity metrics and performance monitoring techniques also shaped the foundation for formulating interview questions.

Finally the interview questions were drafted for data collection purposes. They are found in Appendix C.

3.4.2.5 Preparing interview questions and finalising them

At stage 5, the researcher established steps to be followed in preparing and finalising the interview questions:

- 1) Identify the productivity definitions, factors, metrics and performance monitoring techniques that are of interest to the researcher.
- 2) Prepare a list of participants to be interviewed and make sure that all stakeholders involved in software development are involved in order to obtain a balanced opinion. All the participants involved should be identified beforehand.
- 3) Ensure that all ethical research standards are adhered to, including obtaining an ethical clearance from the university.
- 4) Finalise the interview questions and e-mail a list of topics to be discussed to the participants before the actual interview date.

3.4.2.5.1 Selection of participants

In selecting the participants, consideration was given to people who were part of software development from the feasibility stage up to the final software product to be delivered to the client. The participants thus identified were developers, testers, system analysts, project managers, software development managers and business representatives.

The selection of the organisations for the case studies was based on a review of each organisation profile, company organogram and previous project profiles; to confirm their engagement in projects that have used agile methodologies in software development for at least three years. The researcher conducted an internet reference check (through visiting the company website and any link where the organisation was mentioned online) for organisations that use agile methodologies in South Africa in their software development. Among the nine provinces, two provinces had a higher number of organisations that were involved in software development using one or more of agile software development methodologies (for example, XP, Scrum, etc.). The researcher then selected an organisation in each of the identified two provinces and did a background check to confirm that the organisations chosen were relevant for this study by making telephone calls to the organisations and obtaining confirmation from the relevant software development managers. Communication between the researcher and the managers continued through electronic mail while organising participants and interview dates that suited the management and the development team schedule. Care was taken to identify and select the right participants who were currently involved in an agile software development project. A department organogram was requested by the researcher from each manager representing each organisation to allow the selection of appropriate participants that would be able to give the answers that are relevant to the research study.

With the assistance of each of the selected managers, personal electronic mail invitations were sent out to prospective participants in both organisations. Participants were selected based on their job description and experience with agile methodologies. Caution was taken to ensure that the interview questions were unambiguous and clear, to make sure that the participants would be in a position to answer questions clearly. The participants

who were chosen to participate in the focus group interview were the same participants who were first interviewed as individuals. The focus group interviews were scheduled to last for ninety minutes.

3.4.2.5.2 Ethical considerations

Ethical considerations are very critical in research. Ethics are standards or norms for conduct that differentiates between right and wrong (Munhall 1988:150). The researcher adhered to all ethical standards by first obtaining an ethical clearance from the university, although this research focused on productivity; the people under study were not harmed nor placed at risk, and were treated fairly and with dignity. The right of participants was considered as one of the most important ethical principles. The participants were informed in writing that they had the right to participate or not; they could withdraw at any given time, they had the right to give informed consent, and they had the right to privacy and confidentiality. The names of all the participants in both organisations under study will be kept confidential.

An interview protocol (see Appendix A) was established for use by the researcher in both organisations. The following interview protocol procedures were considered in the preparation of interviews:

- (1) The researcher developed a script that helped the participants to understand their rights, and it ensured that the researcher conducted her interviews in an ethical manner.
- (2) The researcher collected consent from all participants before the interviews began.
- (3) The researcher would inform the participants that she was using recording devices and taking brief notes.

An informed consent statement (see Appendix B) was also considered and contains the following:

- (1) The purpose and procedure of the research, reasons for undertaking the research and the benefits expected from the research.
- (2) The identity of the researcher, supervisors and the university.
- (3) A guarantee that the participants' anonymity would be protected and data obtained from them would be kept confidential.
- (4) A statement of the right to withdraw from participating at any point in time, and that participation was voluntary.
- (5) A statement indicating that there was no incentive or payment involved in participating in the research.

All researchers need to consider and abide by research ethics. All ethical considerations regarding rights of participants and responsibilities of the researcher were addressed for this study. The University of South Africa's main ethical consideration was followed, namely that researchers must apply for ethical clearance before conducting any field research, and have to obtain approval. An ethical clearance was issued to the researcher by the university. A non-disclosure agreement with the participating organisations was signed, to establish a formal understanding between the researcher and the organisations with the guarantee that data confidentiality would be maintained.

3.4.2.5.3 Finalising interview questions

Table 3.7 below gives an overview of productivity factors, mapping them to the interview questions. Appendix C has a list of the interview questions. A semi-structured interview with open-ended questions was used, to allow the order of questions to be changed depending on the flow of the responses the researcher would be getting from the participants.

Table 3.7 Finalising interview questions

	Common software productivity factors	Interview questions
1	Team size	Q2. a; e;
2	Capabilities and experience of the team	Q2. b; e; g; h

	Common software productivity factors	Interview questions
		Q3. a; c; d Q4. e;
3	Complexity of the software	Q2. d; e
4	Constraints in a project	Q2. e; h
5	Usage of tool and quality/effectiveness	Q4. a; d; 6
6	Methods used	Q5. b; c; e
7	Process maturity and stability	Q2. b; c; f;
8	Team motivation and commitment	Q3. a; b; c; d; e
9	Required software quality	Q4. f Q5. c; d
10	Tool usage	Q5. b; e
11	Quality of reusable code	Q2. g
12	Product complexity	Q2. f
13	Product support	Q2. H; Q5. a

3.5 Summary

In this chapter, a detailed discussion of the research methodology in context was presented. A theoretical framework was proposed which guides the research process for this study. The research approach was discussed and qualitative research was chosen for this study. The research philosophy was described; the research lies within the interpretivism paradigm. The research strategy was also presented and discussed in detail. Case studies were chosen as the most appropriate research strategy to be used for this study. Data collection and analysis was also discussed in detail. Individual and focus group interviews were chosen for this study. The research design was discussed and the research process showing the research phases was also presented. The participants' selection and ethical considerations were also discussed for this study.

Chapter 4 shall present the actual data collection and data analysis.

Chapter 4: Data collection

4.1 Introduction

Chapter 3 brought a comprehensive discussion of the nature of research and research methodology. That chapter also addressed the rationale for the choice of research strategy and data generation methods for this research. The present chapter presents a discussion of the data collection method and data analysis used for this research, and also of the procedures used in collecting data.

A pre-data collection process was used as the starting point for data collection. The pre-data collection was a critical part in the data collection process, as it outlined the base for the formulation of interview questions. The second part of data collection was to present the gathered data. The data collected were presented showing factors that affect software productivity in agile teams in companies that adopted these methodologies in South Africa.

An outline of the process used in obtaining these factors and values and/or agile practices is shown below in figure 4.1.

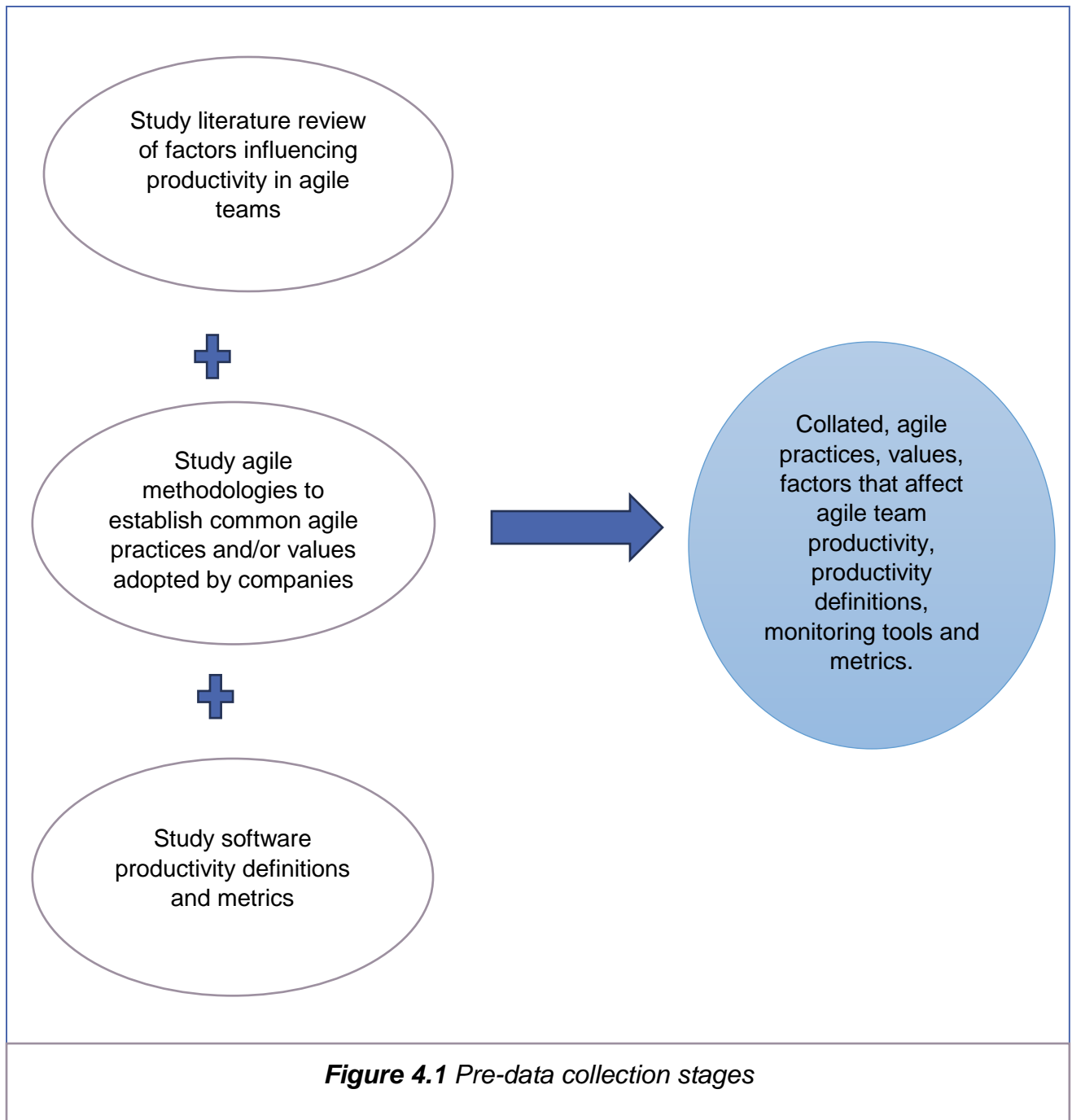


Figure 4.1 Pre-data collection stages

4.2 The interviews

Oates (2009:186), described an interview as a particular discussion between people, with a set of assumptions not applicable to a 'normal' conversations. A semi-structured interview which is theme-based was the main data collection method in this study.

Main themes:-

- The company profile: gives details about the divisions and agile projects on which the teams are working.
- The company software development process: about adopted agile practices and processes, including the practices and processes being implemented in that organisation.
- Team design: about team composition, task structure and team diversity.
- Individual factors: gives an overview of individual skills, abilities and individual member turnover.
- Team environment: about work settings, orientation and stage of team development.
- Task construction: about task design, stage of team development, duration, team independence and interdependency.
- The organisation's perceptions on key issues: how the staff and customers reflect on productivity, software quality, business value creation, customer satisfaction, etc.

4.2.1 Selecting the interviewees

The participants who were selected were the persons with experience and knowledge in agile software development projects (Scrum or Extreme Programming (XP)). Agile teams use roles and not positions, which means any given individual takes on one or more roles at any given time, and roles can be switched over time. The participants' knowledge was acquired through formal education, experience and roles in agile projects as follows:-

- (i) **Team Lead:** - This role can also be referred to as Scrum Master or team coach or project lead (lead programmer). The Team Lead was responsible for obtaining resources for the team, organising the team's work and ensuring that the team is protected from problems. The role of the Team Lead encompasses the soft skills of project management.
- (ii) **Team Member:** - A team member can either be a developer or a programmer. A team member's responsibility is to build, create, develop and deliver the system. This includes software modelling, programming, software testing, and release activities.

- (iii) **Product Owner:** - A Product Owner is also referred to as On-Site Customer in Extreme Programming (XP) and represents the stakeholders. The Product Owner is a person who is responsible for a team's prioritised work item list, decision-making in a timely manner and providing needed information in a timely manner.
- (iv) **Stakeholder:** - A stakeholder is any person who is a direct or indirect user, senior manager, manager of users, operations staff, the project fund owner, support staff, auditor, portfolio/program manager, etc.

Abbreviations such as TL1, TL2,, TL_n were used in the case study to indicate the roles of the Team Leader (for example), and this applies to all the roles under study in agile teams.

4.2.2 The interview procedure

The method used for conducting interviews was mainly face-to-face conversation. Electronic mails were used as a backup to address issues that needed clarification and were not fully presented during the interviews, or any others that were deemed important in relation to interviewees' answers. Eight face-to-face individual interviews and two focus group interviews were conducted. Each face-to-face interview lasted for approximately one hour. Focus-group interviews were conducted after individual interviews to enable rich data that might have been missed in individual interviews to emerge. Focus group interviews allowed interviewees to express ideas that they had missed out during individual interviews. The focus group interview also served to triangulate data from different sources.

To come up with an all-inclusive set of interview questions, the following steps were used as a guideline:-

- The literature was studied thoroughly to ensure that the research study included the latest advances.
- To validate the list of factors that influence agile team productivity, expert analysis was used. As these factors form the foundation for the research interview questions, care was

taken to ensure that factors affecting software productivity in agile teams covered all aspects of this research.

- Interviews were booked and confirmed via electronic mails, and travelling arrangements were made prior to interview dates. Four sets of interviews were conducted in Gauteng Province, covering all the aspects of factors that affect software development productivity in agile teams. The interviews were conducted in two separate days, but in the same location. Four sets of interviews were also conducted in Western Cape Province, all within one day.

4.2.3 Conducting the interviews

The purpose of the interview was explained by the researcher to each and every participant, and in some cases to the participants' line manager. This information was conveyed to the participant at the start of every interview, and the reason for selecting that participant was explained. Participants were e-mailed a written-consent form before the interview date, and this was then signed by the participant in the presence of the researcher before commencement of the interview. The estimated duration time of the interview (which was approximately one hour), data confidentiality relating to data collected during in the interview, and the use of audio recording and note-taking were fully explained before the commencement of the interview.

The companies selected for the interviews are competitors in the software development industry in South Africa; thus care was taken to guarantee that participants felt comfortable in answering the interview questions. The first phase of data gathering was conducted in Gauteng Province, and it was based on a list of interview questions formulated in advance. The data were collected using face-to-face interviews in Gauteng Province, and the responses and emerging data gathered in the first interview assisted the researcher to rephrase some of the interview questions in this early stage of investigation. The researcher reviewed the questions, and some were then asked in a different way and/or modified to be more meaningful to the participants. Verification of information was carried out where necessary. The second interviews were conducted in Gauteng Province in the same organisation but with different participants. The final set of interviews was conducted in Western Cape Province. A full day was set aside for these

interviews in Western Cape Province since the researcher had to travel to conduct the interviews.

4.2.4 The interviews – Process

A theme-based interview was the main data collection method. All interviews were audio recorded and then later transcribed. The recorded interviews gave the researcher a good opportunity to review the participants' responses, the tone of their voices and the content of what was said. Seeking clarification and asking follow-up questions was possible during the face-to-face interviews, which resulted in the researcher obtaining rich data. E-mails were used as a backup to address issues that needed clarification and were not fully presented during the interviews, or any others arising from the answers.

In order to familiarise herself with the collected data and remember the interviewees' responses, the researcher listened to every digital recording immediately after each interview and made notes of each interview. The researcher also took notes of her observations before and during the interview time. For instance, the interview time set was at 10:00 am but the participant became only available at 10:30 am. The researcher took note of these observations, as time management has a direct effect on software productivity. The researcher then transcribed and noted down relevant points on the audio recordings. Every transcription was read and further notes were made. After completing the initial reading, another reading was done by the researcher while listening to the audio recording. This was done in order to take note of recurring words, concepts, ideas and themes in the data collection process.

4.3 Data analysis

Data collection and data analysis were conducted concurrently. According to Yin (2002:111), "there must be an analytical strategy that will lead to a conclusion." Three strategies were presented by Yin (2002:111-115) for general use:

- The researcher should rely on theoretical propositions of the research, and then he/she can analyse the evidence based on the propositions.

- The researcher can create a case description that should be a basis for organising the case study.
- The researcher can establish a framework based on the rival explanations.

The general strategy used in this study is the case description one, where the descriptions were those that resulted from the literature review in the previous steps (see Chapter 2) and were reflected in the case study (see interview questionnaire Appendix C). Hence for this study, the analysis was conducted concurrently with the data collection phase. A sample of data was collected and analysed regarding factors that affect agile team software productivity, as depicted in table 4.4. Participants in this research had experience of working in agile projects for three years and above. Face-to-face interviews were used and helped the researcher to explain terms that the participants were not familiar with. For this research, data were coded and analysed by the researcher according to the significant factors affecting software productivity in agile teams. The samples of data collected and analysed are presented in Appendix D. The collected data were then purified based on the outcomes or results of that data analysis.

As part of analysing the data gathered, statements and comments from participants were provided in Section 4.5 to help coding and categorising the collected data for a better understanding. The researcher ensured that no participant's name or organisation's identity was disclosed. The codes used to describe the participants were TL1, P1, PO1, PM1, etc. for both Gauteng Province and Western Cape Province participants. Some statements and/or information that might have revealed the identity of the participant were edited to maintain confidentiality.

This research falls under the description of exploratory study. The issue of an appropriate level of analysis needed to be addressed, and the researcher decided to use qualitative content analysis as it adheres to the naturalistic paradigm and is a research method well suited for the subjective interpretation of text data content through a systematic classification process of coding and categorising patterns and themes (Hsien & Shannon, 2005:1278). Incorporation of content analysis in this study was done at the pre-test stage during the development of the interview guide and was used as a foundation for the coding scheme, and for evaluating the

effectiveness of certain interview items. The data collection in this study formed the foundation of data analysis. Data analysis comprises three simultaneous flows of activity, namely, data reduction, data display, and conclusion-drawing (Miles & Huberman, 1994:2). These activities are discussed in detail below:-

4.3.1 Data reduction

Data reduction was used to reduce and condense data as the study progressed. Reduction of the data helped to compare, contrast, aggregate, sort and order data in order to allow for final conclusions. The data reduction process began with a focus on distilling what the participants reported about the practice, activity or phenomenon under study, for knowledge to be shared. Based on the research questions and problem statement, the data reduction process was used to obtain data that were applicable to this study. The data reduction and condensation was carried out within the boundaries of qualitative content analysis by using a coding process. Irrelevant data collected were thrown out since they did not have any relevance to the study and participants did not have exposure or experience that enabled them to answer such questions.

4.3.1.1 Qualitative content analysis

Hsein & Shannon (2005:1278), defined qualitative content analysis as a method of research that uses subjective interpretation text data content through the systematic classification process of coding and categorising patterns and themes. Content analysis is today's most frequently employed analytical tool and has been fruitfully used in most research applications in library and information sciences (Allen & Reser, 1990:251). As content analysis is a naturalistic paradigm, the participants' responses were interpreted in such a way as not to compromise their original versions. For this study, content analysis was chosen by the researcher because of its power to make faithful inferences. The research purpose was to identify factors that influence software productivity in agile teams. According to Schamber (2000:739), a coding unit is a term or group of terms that could be coded under a single criterion category. The researcher unitised responses to each interview before they were coded. A sequence of codes and categories were developed by the researcher for content

analysis as patterns emerged. The content analysis findings were presented in tables and diagrams.

4.3.1.2 Coding

Coding was done initially in the study. It enabled the researcher to start scanning recorded data and developing categories of phenomena. Codes were used by the researcher as a way of starting to obtain the meaning of the data. According to Taylor & Gibbs (2010:234), coding the data makes it easier to examine the data, to make evaluations and detect any patterns that require to be investigated further.

For this study, coding took place in several stages over time. Open coding was the initial coding process, where the researcher thoroughly read and noted every interview script. The texts were unitised, concepts were highlighted and labelled during the open coding process. Subsequent coding took place to constantly compare the current transcript with preceding ones, allowing new categories and their properties to emerge. During the coding process, further activities and themes emerged. A list of factors that affect software productivity in agile teams and coding were defined and listed in Table 4.1.

4.3.2 Data display

Data display is the next main flow of data analysis activity. It affords a compressed, organised assembly of data that allows conclusions to be drawn (Miles & Huberman 1994:235). Data display allows for the extrapolation from data, enough to permit the researcher to start to identify any systematic interrelationships and patterns. At this stage, additional themes and categories are likely to emerge from the data, superseding those initially discovered during the first data reduction process. Therefore, data could be displayed using, for instance, flow charts that allow the mapping-out of any critical path or decision points.

For this study, data were displayed through paragraphs of texts, tables, flow charts and diagrams.

4.3.3 Conclusion-drawing and verification

Conclusion-drawing begins by the researcher noting explanations, regularities, propositions, causal flows, possible configurations and patterns (similarities/differences). The researcher should review the meaning of the analysed data and evaluate any implications for questions that need to be answered. According to Miles & Huberman (1994:235), the meaning emerging from the data should be verified for sturdiness, plausibility, and confirmability – that is, data validity.

For this dissertation, the data were revisited many times as necessary so as to cross-check and verify the emergent conclusions.

4.4 Data collection foundation – Factors that affect agile team productivity

The following section discusses steps that were used in gathering information which forms the basis for compiling interview questions.

According to Dey (1993:258), for the researcher to produce an ‘acceptable’ account of the study, some general criteria have to be met. The three standard criteria for any analytical work were identified as follows:-

1. Reliable – the findings should be consistent.
2. Valid – the findings should tell the correct information.
3. Representative – the study should share the same findings.

The essence of reliability through consistency was tested by repeating the same interview questions to different participants located in different geographical areas. To check the reliability of the case studies, interviews were conducted on separate days with different participants working in the same company. Internal replication was undertaken to test the reliability of the researcher’s analysis.

A valid account is defined as one “which can be defended as sound because it is well-grounded conceptually and empirically” (Dey, 1993:258). Validity was achieved by carefully considering the quality of information sources, and cross-referencing the case studies. The interviewees’ responses from the initial interviews were cross-referenced with the responses from successive participants in order to confirm data consistency. Any new information that arose was noted for further internal validation. External validation was carried out through the review by experts in agile methodologies.

4.4.1 Collating agile productivity factors

Using factors obtained from the researcher’s literature review in Chapter 2, an expert analysis was conducted to find out which agile factors affect team productivity in the companies under study, and whether the factors effect team productivity in a positive or negative way. This preliminary analysis was an important process as it formed the foundation for identifying common factors that affect agile team productivity in both companies under study. A comprehensive list of these factors was compiled, as in Table 4.1 below:

Table 4.1 Collated agile productivity factors and description

Item	Productivity factors	Brief description
1.	Team morale	Projects have to be built around motivated developers.
2.	Team collaboration	Working together jointly as a team, sharing knowledge, experience and skills.
3.	Continuous integration	Involves continuous building of the project as soon as there are any changes.
4.	Daily meetings	A daily morning fifteen minutes stand-up, short meeting for the agile team to sync.
5.	Size of database	The database should be completely flexible and evolvable. The database should evolve as an application develops.
6.	Product complexity	The size of the software product.

Item	Productivity factors	Brief description
7.	Code reusability	Using existing code to build new software.
8.	Software tools used	The tools used to enable the team member to execute their jobs.
9.	Platform instability	Software written for a certain platform has very little use when transferred to other platforms.
10.	Capability of programmers	Constant attention to good design and technical excellence improves agility.
11.	Employee turnover	Number of employees leaving the team over a period of 12 months.
12.	Experience with different applications	Team members should learn new skills from one another because it is more valuable to the organisation and can be better equipped in supporting each other's' work.
13.	Experience with using language and tools in use	The team experience with different programming languages and tools.
14.	Multisite development	Software development teams in different geographical locations collaborate on common software projects.
15.	Required development schedule	Working software should be delivered on a frequent basis, maintaining a shorter timescale.
16.	Real customer involvement	Customers and developers working together throughout the project on a daily basis.
17.	Planning meetings	Planning meetings that are held by the team at the beginning of each iteration, generally lasting from two to four hours.
18.	Root cause analysis	Used to facilitate a shared understanding of problems and stopping those problems which may been bugging the team for a long time.
19.	Refactoring	The activity of refining the design code without changing the external behaviour.
20.	Self-organising	Typically the team members do not require any supervision or direction.

Item	Productivity factors	Brief description
21.	Team size	The number of team members per group.
22.	Management quality and style	Management are prepared to do things differently and try new things, quality assurance is seen as a shared responsibility.
23.	Project constraints	The developers can get the job done, if a good environment and support is provided.
24.	Required software quality	The key measure of progress is quality working software.
25.	Open communication	Face-to-face communication is the most effective and efficient way of transmitting information within the team.
26.	Transparency	Openness within the team members and the business people.
27.	Risk taking	Risk is seen as inevitable but also as an opportunity by the team.
28.	Responding to change	Changes in requirements should be accepted at any time

The above factors which affect agile team productivity were maintained and used as a basis for research questions.

4.4.2 Validating agile productivity factors

To confirm the validity of the above factors, the input from three agile experts who were working for the companies under study was incorporated. The experts were selected based on the following:-

- The first expert had been working for one of the organisations under study for more than six years starting from a junior position, was now in middle management and had worked on agile projects for the past six years.
- The second expert holds a senior role in the agile team and has extensive experience working on agile projects.
- The third expert was involved in both software development using agile methodologies and managing a team.

The feedback obtained from the experts was analysed by the researcher as well as their rating in terms of each factor effect on productivity. The agile experts' comments were then reviewed thoroughly. The feedback provided assisted in further clarifying the meaning of the collated factors as they form the foundation for interview questions.

4.5 Data collection

Using Kitchenham et al. (2008:101) - guidelines provided for conducting and reporting case studies research - Table 4.2 provides a summary of the profiles of the companies under study. Each company's agile adoption level in their projects is shown in Table 4.3. The adoption level was assigned as full if the company is adopting that practice fully, partial if the company is using the practice sometimes and do not use if the company does not use the practice at all.

Table 4.2 Summary of project and company profiles

Characteristics	Company X	Company Y
Company Activities/ business type	Software development for financial institutions.	Financial software solutions provider.
Organisational structure	Horizontal	Vertical
Total number of IT staff	45 (50 company staff complement)	32 (142 company staff complement)
Description of project	Financial system, with calculation engine, fees engine, commission management, and single sign on.	Payments and collections processing.
Team build up	6 - Team members 2 - Senior developers 2 - Intermediate developers 1 - Junior developer 1 - Manager	7 - Team members The biggest team composition was 7, comprising developers from senior to junior level and a Product Owner.

Characteristics	Company X	Company Y
Programming languages	Java, C#, SQL	C#, JavaScript, html, CSS, SQL, MONGODB
Requirements stability	Initially low but increases with time to medium.	Low
Non-functional requirements	Client relationship management. Ownership of work and of the project, Reliability	Availability, reliability and performance
Reuse	Components	Components, in-house and 3 rd party
Percentage of staff turnover	5 % per annum	1,25 % per year

Table 4.3 Extreme Programming and Scrum practices adopted by the companies

Practices/ Values	Company X			Company Y		
	Full	Partial	Do Not Use	Full	Partial	Do Not Use
Refactoring	✓			✓		
Code and tests	✓			✓		
Small releases	✓			✓		
Continuous integration		✓		✓		
Collective ownership		✓			✓	
Daily meeting	✓			✓		
Energized work	✓					✓
Daily deployment	✓			✓		
Incremental design		✓		✓		
Onsite customer		✓			✓	
Pair programming		✓			✓	
Shared code		✓		✓		

Practices/ Values	Company X			Company Y		
	Full	Partial	Do Not Use	Full	Partial	Do Not Use
Sit together	✓			✓		✓
Single code base	✓			✓		
40 hour week		✓		✓		
Test driven development	✓				✓	
Negotiated scope contract		✓			✓	
Ten minute build	✓			✓		
Planning game		✓			✓	
Root cause analysis		✓		✓		
Informative workspace		✓		✓		
Stories	✓			✓		
Slack	✓				✓	
Weekly cycle (every two weeks)		✓		✓		
Team continuity	✓					✓
Whole team	✓					✓
Retrospectives	✓				✓	
Knowledge worker		✓				✓

4.5.1 Data Collection – Interviews

The following section contains detailed information that was collected during the interviews. Interviews were conducted with eight team members within the two companies, including team lead, developers, product owners, and project managers, considering their experience and roles.

The interview questions were grouped into five categories to allow the researcher to answer the research questions, as follows:

1. Individual and/or group questions
2. Team design questions
3. Individual and team factors questions

4. Time management questions
5. Communication strategy questions

As the base for data presentation, agile productivity factors and coding were used.

As noted in the previous chapters, the purpose of this study is to develop a framework for understanding the factors that influence software productivity in agile teams. Thus there is the possibility that certain factors have a positive, negative or neutral influence on some of the identified productivity factors.

This section looks at the data collected as an initial review and then the data are presented in an atomic way of factors that affect the productivity of agile teams.

The symbols below were used to indicate the meaning of the outcomes:

(+) shows that the factor has a positive influence on productivity

(-) shows that the factor has a negative influence on productivity

(+/-) shows that the factor has a neutral influence on productivity

() shows that the factor was not mentioned or was not applicable during data collection in that company

4.5.2 Data collection – Importance of adopting agile methodologies

This section deals with the relevance of the collected data for companies in South Africa that are considering the adoption of agile methodologies.

4.5.2.1 Company X

Data collection revealed that by adopting agile methodologies i.e. Scrum, the companies have reaped a lot of benefits by focusing on the development of only the required items. They are able to prioritise their work, embrace change, have a closer working relationship with their clients, faster turnaround time, frequent deliveries, and the agile methodology is less stressful on their developers.

The following comments were made during the interviews:

- *'the biggest benefit with agile methodologies is getting users involved and delivering something to users, even if it is small, and they can start fiddling around with it (participant P1).'*
- *'Instead of all aspects being equally important, the client is required to prioritise, avoiding the development of unnecessary items (participant TL1).'*

The comments indicated that agile methodologies are beneficial in this company as only the required product is developed and the clients' approval at each stage eliminates unnecessary items that might not be useful to them.

4.5.2.2 Company Y

From the data collected in this company, the company uses disciplines in Extreme Programming and some part of Scrum. The company has benefited from adopting agile methodologies since now they are able to manage their technical debt down to a reasonable level. The benefits noticed were small cycles, small feedback loops, early release and lower development cost because they found out that they could notice problems earlier and fix them before they pan out to become more problematic. The following statement clearly shows that there are visible benefits obtained from adopting agile methodologies in this company:

- *'Well, I suppose the main thing is our technical debt is between fifteen to thirty percent on most of our projects, and our clients' technical debt amounts between seventy to one hundred percent and sometimes more (participant LP1).'*

After adopting agile methodologies, the company found that they were now able to run their technical debt on their projects lower than their competitors and clients, which of course was beneficial to the company.

4.5.3 Data collection – Factors that influence productivity in agile teams

Data collection in this section covers the factors that affect software productivity, and their effect on agile teams.

4.5.3.1 Company X

The data collection from this company revealed that factors that influence their team productivity were both negative and positive. The following section discusses the factors using the symbols presented in section 4.5.1 to represent the outcome of each factor.

(+/-) Process – Agile Processes: The data-collection information indicated an inefficient process, where an agile methodology was obviously abused or used incorrectly, affected productivity in a negative way. Discussions revealed that where there was an inefficient process, a lot of time was wasted developing or fixing unnecessary bugs.

The following comments were mentioned during the interviews:

- *‘when your work hasn’t been scoped correctly, there is a lot of unknowns before you go into it, so you end up doing work, and you get to it, and see that there is far more than what you realise (participant TL1).’*
- *‘Red tape is a problem, dependencies make the process slow down (participant P2).’*

The information gathered with regard to ‘process’ was that if the agile process is followed correctly, processes are working well and interactions with clients are going well, productivity within the agile team can improve. The statement *‘following processes helps with productivity, doing your grooming sessions, have retrospectives, planning properly, has a very big role in being productive (participant P1).’*

(-) Knowledge – Team member skills: The responses gathered from the interviews revealed that knowledge is always an issue when working on a project because the skills of team members and how quickly they actually catch up on new technology and languages is the key issue which affects productivity. Knowledge workers are part of agile teams, sharing the same goal and working together. Knowledge workers’ productivity is typically intangible. Participant TL2 points out that: *‘knowledge of your team members is critical when working on a project (participant TL2)’*, clearly indicated that in agile teams, knowledge is perceived important in team productivity. If team members are knowledgeable, productivity increases but if they lack the knowledge, productivity level goes down. Lack of knowledge among team members had an effect on productivity in this

company's teams and most participants agreed that training was needed to enhance the skills of most developers.

(+/-) **Team morale - Team motivation and commitment:** The data-collection information indicated that if the team gels well, processes are working well and interactions with the client are going well, productivity is higher due to quicker turnaround time and quicker feedback from clients. The statements '*if the team gels well and there are good lines of communication, your productivity goes up (participant TL1)*', '*doing grooming sessions, has a very big role in being productive (participant TL2)*.' These statements clearly showed that if the team worked together, were committed to their work, and were highly motivated, the productivity level increased. However, if team morale went down, productivity also went down because software is developed by people and if the people are not motivated to do their jobs, they will spend their time doing non-productive things. The participants listed team morale as one of the top factors that influence productivity in their company. When there was no trust, there were bad lines of communication, wrong impressions from people and insufficient processes; all of these highly affected team morale, with negative effect on this company's team productivity.

(+/-) **Management style and quality – Management support:** The participants revealed that there was support from management but the team itself was essentially responsible for deciding how things should be done. The responses from participants indicated that management support was being offered, but surely additional management support was required in certain areas. Participant TL1 and P2 were quoted saying: '*when there is a problem, they are there to help out, smooth out the process (participant TL1)*', '*in most scenarios we are motivated and encouraged to find our own solutions and be innovative (participant P2)*', indicating that management support was there but can clearly be improved by management getting involved and showing interest in the work done by the team. Hence management style and quality was seen as having both a positive and negative effect on team productivity. If management support increased, productivity increased and with minimum support from management, productivity level reduced.

(+/-) **Team size – Number of team members per project:** The responses gathered from participants indicated that most agile teams in this company had from three to six programmers, ranging from senior to junior level. The data-collection information indicated that team size or having enough team members affects productivity in both positive and negative ways. If team members were adequate for the assigned project, the productivity level went up, but if there weren't enough team members for that given project, the productivity level went down. The statement acknowledging the effect of team size was quoted from participant P2: *'in the project that I am currently working on, I get pulled away out of the project to help another team get their project delivered (participant P2).'* The gathered information indicated that some teams did not have adequate resources (team members), so they use team members from another group which rather had a negative effect on productivity for both teams in question. However, some teams which did not have such problems and had enough team members, agreed that their productivity level improved as the team was maintained for the duration of the project.

(-) **Team collaboration – Teamwork:** Team collaboration was there in general but often other team members were not aware all the time what their colleagues were doing. Sometimes other team members would find out that the problem was solved but there were not even aware of it. Participant TL1 points out that: *'because it's a lean organisation, management makes decisions very quickly and team members don't have enough time to know about it (participant TL1).'* The information collected clearly indicated that team members do not inform each other of what was happening and this had a negative effect on productivity since two developers might be trying to solve the same problem at the same time but sitting in different locations from each other.

(+) **Daily meetings – Daily stand-ups:** Daily ten-minute stand-ups had a positive influence on this company's team productivity. The daily stand-ups assisted the team members to deal with rising temper as during the meetings, there was one person who would lead the discussion and assured the team that they would deliver the project successfully. Figure 1.1 below shows developers having their daily stand-up meeting.



Figure 4.2 Team at an early-morning daily stand-up.

(+) **Continuous integration – Response to change:** The data gathered indicated that company X responded to changes very quickly. Participant P1 was quoted confirming that: *‘one of our strong points that we have is that we are able to adapt and are able to change (participant P1)’*. Responding to change and continually building the project according to clients’ changes in requirements was seen as one of the factors that have a positive effect on team productivity. The following statement is quoted from participant TL2: *‘we just experienced that, if one of our system has a flaw or it was missing a big piece and we had to refactor it and it was done exactly in the next sprint (participant TL2)’*. The data collection information indicated that turnaround on refactoring and implementing changes was very short.

() **Size of database – Size of features developed:** This aspect was not discussed with participants from company X.

(+/-) **Complexity of the product – Product/ Project complexity:** The data-collection information indicated that product complexity had both positive and negative effects on productivity. Participant TL2 pointed out that: *‘some things are less complex but take a lot*

of time, other things are small to do but there is complexity around actually trying to figure out how you are going to do it (participant TL2)'. If there were requirements that the development team were not aware of, it made it difficult for them to “size the stories”. If the stories had the same sizing, they immediately knew that they were able to do x amount of story points in one “sprint” (two weeks). If the size of the product was known, it clearly had a positive effect on productivity because the team was able to plan properly per each sprint - and vice versa.

() **Code reusability – Code reuse:** This aspect was not discussed with participants from company X.

() **Software tools used – Programming tools:** This aspect was mentioned in passing by company X participants. The data-collection information indicated that if programmers have the tools they need in place, they can be productive.

(-) **Employee turnover – Team member turnover:** The data collected indicated that employee turnover affects productivity in a negative way. If more than one developer left in the middle of the project, it affected productivity because the new developers required training in order to catch up with the existing team. The company employee turnover rate for the past year was five percent which was higher considering the size of their teams. At least two developers were likely to leave the company in each team per annum.

(+/-) **Multisite development – Geographical location:** Company X deployed all of its development team to client sites. Software development was not carried out at one site but at a number of client sites. Multisite development was seen as having both negative and positive effects on productivity. A positive influence was that the programmers were working close to the client and that they were doing work with the client fully involved. A negative aspect is that at times the project being worked on placed the programmers in different locations, as shown for example by the statement *‘in the previous project, the client was based in Cape Town and they were not engaging as they should because we were using their Johannesburg branch whilst the software was built for the Cape Town*

branch (participant TL1)'. The data-collection information therefore indicated that if there was not enough engagement and clear communication with team members working on the same project, but located in a different geographical area, productivity went down.

(-) **Required development schedule – Duration / Deadlines:** The information collected indicated that the clients managed their schedule themselves and their schedule did not have any effect on the productivity of the team of Company X. Participant TL2 points out that *'client is in control of the development direction (participant TL2)*', clearly indicated that the customer was responsible for managing the duration of the project.

(+/-) **Real customer involvement – Onsite customer:** The collected data indicated that involving customers in software development had both positive and negative effects on productivity. In projects where clients were involved, there were faster turnaround times which had a positive effect on team productivity. To quote participant TL2: *'we have weekly sessions with our clients showing them what functionality we have developed. We physically show them new screens, work through them with the screens, get their feedback on how they have experienced with the screen and it had been working brilliantly (participant TL2)*', clearly indicated that customer involvement had a positive effect on team productivity. In cases where clients were not involved, productivity went down because the programmers developed software that, at the end of the day, could not meet client needs. Participant TL1 acknowledges that: *'we get some clients where they don't get involved too well (participant TL1)*'. The data collection indicated that the client would partially get involved, disappeared and when they came back, they often realised that a lot of things had changed at a later stage of development, which created problems for the development team when they had to rework items that the client was not happy with.

(+) **Planning meetings – Weekly or bi-weekly meetings:** The data-collection information indicated that planning has a positive effect on team productivity. Planning properly played a big role in the team for it to become productive. Participant TL2 was quoted confirming that: *'we have actually noticed that if you don't plan properly, your productivity goes down, because misunderstanding functionality can cause you to*

develop the wrong solution (participant TL2)'. The information gathered clearly indicated that planning if done properly, had a positive effect on productivity. Figure 4.2 below shows the team planning for the next sprint:



Figure 4.3 Team sprint planning activities

(-) **Root cause analysis:** This aspect was not discussed with participants of company X.

(+) **Refactoring:** In this company, MVP (Minimum Viable Product) was used to keep the refactoring smaller. The data-collection information revealed that MVP enabled them to gather all the requirements for the product before they could start development process. Participant TL2 confirms that: *'turnarounds on refactoring and implementing the changes are very short (participant TL2)*'. The data collection clearly indicated that refactoring impacted team productivity in a positive way in this company.

(+) **Self-organising:** The team was self-organising according to the data-collection information. The developers communicated directly with the clients and interacted with them during the development stage to understand their needs. Participant P2 was quoted

saying: *'the users haven't communicated with the developer before but since we introduced agile to our clients, users can now interact with developers (participant P2)'*. The team itself would go out there and gather the requirements by themselves from users, i.e. clients, and started coding.

(-) **Project constraints:** Project constraints had a negative effect on team productivity, according to the data-collection information, because if there were constraints in the project, for instance, lack of client input during development stage, the project slowed down and at times such projects end up not being able to be delivered at all.

(+/-) **Work environment – Work setting:** The work setting in this company was different for each team. Some teams were permanently stationed at a client site and some teams were working from the company's premises. The data collected indicated that work environment could have a positive or negative influence on team productivity. The following statements were quoted from participant TL2: *'at the client site where I am working, it's quite noisy and they have been doing renovation lately. We always get moved from one place to the other, which affects our daily productivity (participant TL2)'*. A noisy environment was considered having a negative effect on team productivity and a quiet environment the opposite. Figure 4.3 below shows the work setting for developers.



Figure 4.4 Developers work setting (Company X)

4.5.3.1 Company Y

The data collection in this company revealed that factors that influence their team productivity were both negative and positive. The following section discusses the factors using the symbols presented in section 4.5.1 to represent the outcome of each factor.

(+/-) Process – Agile processes: The data-collection information gathered indicated that processes used in the organisation affect productivity both positively and negatively. If the processes were followed properly, the team productivity went up but if they were not followed properly, the team productivity went down because programmers spent most of their time going through unnecessary stuff or fixing bugs. The company used an iterative process for all their projects.

(+/-) Knowledge – Team member skills: The knowledge of team members affects productivity in both positive and negative ways in this company. The-data collection

indicated that for team productivity to increase, developers' knowledge of different applications was very important. The following comment came from participant LP1: *'inexperienced training of developers effects negatively on productivity (participant LP1)'*, clearly indicated that if there are inadequate skills among team members, productivity is reduced because developers spent their time trying to figure out the solution. However, with senior developers in every team, knowledge was transferred to junior developers through grooming sessions, which rather has a positive influence on their team productivity.

(+/-) Team morale - Team motivation and commitment: The data-collection information indicated that team morale was very important in this company's software development teams. Participant LP2 was quoted as saying: *'team morale is one of the factors that have a big effect on our team productivity (participant LP2)'*, *'team morale is everything in software, organisations or IT departments that always have poor morale have a high level of technical debt (participant LP1)'*. The interview responses indicated that developers needed to be empowered by being given the tools that they needed, and the programming processes that they needed to follow, thus ensuring that they can be productive. Therefore, when the team morale was higher, productivity was also higher, but if the team morale was low, technical debt increased which affected team productivity negatively.

(+/-) Management style and quality – Management support: The data-collection information gathered indicated that there were strict lines of communication that needed to be followed between management and the development team. The comments quoted from participant LP1 were: *'all communication in this company goes through the Product Owner (PO), who will then liaise with management or the team (participant LP1)'*; *'management gives instructions with one voice (participant LP1)'*; and *'employees are allowed to be creative and innovative in a constrained type of way and not in sort of a free type of way (participant LP2)'*. The responses obtained showed that although there was some kind of management support offered through the PO, team members did not have direct access to management to air their views. Management style in this organisation affected productivity of the team in both negative and positive ways.

(+) **Team size – Number of team members per project:** The Company had on most of their projects two to seven team members per project, depending on the size of each project. The data-collection information gathered indicated that each team comprised five programmers, one Product Owner and one junior Product Owner. There were six permanent members per each team for the products that were being worked on. The team size was reduced depending on the level of development phase, sometimes if their products were in the maintenance phase, they could only assign one or two developers. The team size was viewed by participants as having a positive influence on team productivity as there were always enough team members assigned per each product, and well-defined responsibilities. Participant LP1 points out that: *'there is a PO, so the person doesn't code, that person liaises with all stakeholders, writes stories and acceptance testing, signs off and does everything regarding product ownership (participant LP1)'; 'one lead programmer on a team and a number of programmers that range from junior to medium and senior (participant LP2)'*. The data collected clearly indicated that in this company, team size has a positive influence on their team productivity because there were no incidents where team members were moved between projects to cover up for inadequate team members in other groups.

(+) **Team collaboration – Teamwork:** Team cohesiveness in this company had a positive effect on team productivity, according to the data collection gathered from participants. The team worked together to solve problems and take full responsibility of any major impact as a team. A visible project matrix was used in this company where any stakeholder, user, manager or consultant would come in at any point to look at the scramble of the stories and see which ones have high priority. Participant LP2 was quoted saying: *'the team's work is completely transparent to all stakeholders (participant LP2)'*, indicated that teamwork was the centre of their team productivity. Most participants revealed that teamwork in agile teams required a lot of collaboration and communication. Figure 4.4 below shows developers communicating with each other during the development session.

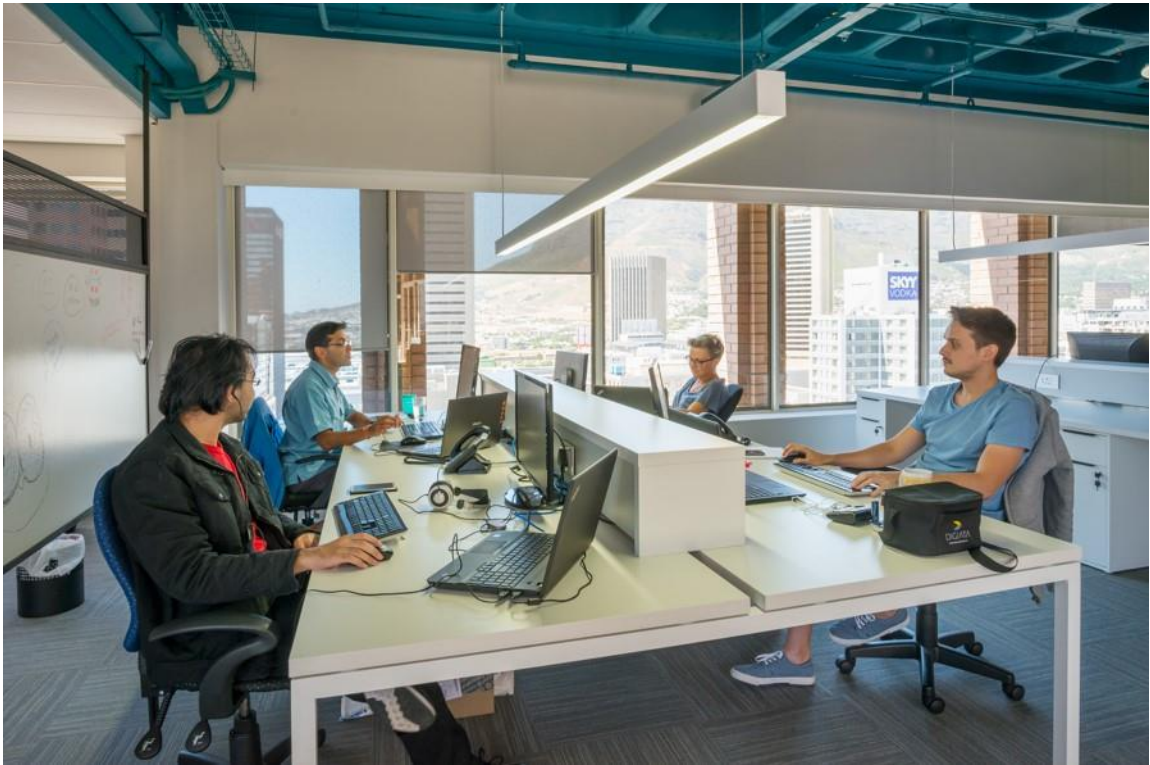


Figure 4.5 Agile team having an informal discussion

(+) **Daily meetings – Daily stand-ups:** The daily stand-ups, short five-minute meetings, had a positive effect on team productivity in this company. Developers would discuss the day's activities and brief each other on the progress of their work. The data-collection information revealed that when the team met every morning, discussed the day's activities and work on the most prioritised work, productivity of the team increased because the developers were fully aware of what was expected of them.

(+) **Continuous integration – Response to change:** The data gathered indicated that continuous integration in this company made the feedback loop shorter. When feedback loops were shorter, productivity levels went up since the technical debt was reduced. In most instances, client feedback was obtained early and the programmers would immediately fix the bugs which reduced the company cost to fixing those bugs. The comments quoted from participant LP2 were: *'We can change our current feature set, we can refactor our code to work with the current features. I think that's the difference with us. I think anybody out there will be hard pressed to beat us at change management.'*

That's what we do. That's kind of what we are known for (participant LP2)'. The comment clearly indicated that continuous integration had a positive effect on team productivity in this company.

(+) **Size of database – Size of features developed:** The data-collection information indicated that the database size or structure had a positive effect on productivity. The statement '*we can change our database structures at any point in time and deploy them (participant LP1)*', indicated that despite the size of the database, the company was able to maintain its high level of productivity.

(-) **Product complexity - Complexity of the product:** Product complexity affected productivity in a negative way in this company. The data-collection information indicated that client confusion about specifications, i.e., using scenarios where there were too many unknowns, affected productivity in a negative way. The statement quoted from participant P1 was: '*just the complexity makes technical debt increase in complex projects that need integration tools (participant P1)*'. The gathered data indicated that complexity of a product had a negative effect on this company's team productivity.

(+) **Code reusability – Code reuse:** The data collection information indicated that code reuse had a positive effect on team productivity in this company. Participant LP2 was quoted saying: '*we can change our current feature set, we can refactor our code to work with the current features (participant LP2)*'. The interview responses indicated that code reuse was one of the agile practices that enhanced team productivity in this company.

(+/-) **Software tools used – Programming tools:** The tools needed by programmers to get their job done had both positive and negative effect on team productivity in this company. The interview responses revealed that when the team was given tools they needed to do a great job, keeping their lives organised, and taking all process smells out of the picture, productivity was higher. The following statement was quoted from participant LP1: '*the motivational factors are the actual process, where you give the*

developers the tools they need to do a great job (participant LP1)'. Inadequate tools were indicated as having a negative effect on team productivity.

(+/-) **Employee turnover – Team member turnover:** Team-member turnover in this company had both positive and negative effect on productivity. The interview responses clearly indicated that when the team member turnover was high, the productivity of the team was affected in a negative way because the company constantly had to recruit and train new team members, who in most scenarios took time to settle in. When team member turnover diminished - as the company witnessed in the past two years - productivity of the team went up because the team skills and abilities improved as the same team members continued to work on different projects. Participant LP2 acknowledges that: *'our staff turnover rate is 1, 25 % per year, which means only one or two people leaving this organisation in a year for the past two years (participant LP2)*', indicating that when the team member turnover was low, productivity increased.

(+) **Multisite development – Geographical location:** The data collection information indicated that multisite development had a positive effect on team productivity in this company. The company's software development team was distributed over five geographical areas, namely, Cape Town (South Africa), Johannesburg (South Africa), Serbia, Mauritius and India. Multisite development had a positive effect on team productivity in this company because the development teams were positioned closer to client sites.

() **Required development schedule:** The responses from participants indicated that the clients managed their schedule themselves, and their schedule did not have any effect on the Company Y team productivity. The company was not using any timelines, durations or deadlines in developing its products. Participant LP1 was quoted saying: *'the client decides how much time it will take to complete the product (participant LP1)*', clearly revealed that the clients were responsible for managing the duration of their projects.

(-) **Real customer involvement – On-site customer:** The data-collection information indicated that there was no direct communication established between the customer and the developers. Participant LP2 was quoted insisting that: *'you have to have a role that facilitates the communication, that activity will never happen efficiently by itself (participant LP2)'*, clearly indicated that there were absolutely no direct lines of communication between the customer and the developers, which had a negative effect on team productivity since the agile approach emphasises customer-and-developer communication and collaboration. Most participants in senior roles revealed that getting clients to communicate with developers directly had a negative effect on their team productivity because of the time spent by developers trying to explain the technicalities that the users might not understand.

(+) **Planning meetings – Weekly or bi-weekly:** The participants' responses indicated that planning had a positive effect on team productivity in this company. The company held two official meetings: short five-minute stand-ups every day, and iteration meetings. Iteration meetings only happened every one or two weeks, at the beginning of each iteration. The team also held impromptu meetings at any time. The following statement was quoted from participant P1: *'all of that kind of collaboration, discussion is all valuable stuff (participant P1)'*, indicated that when the team discussed the specifications and scope before an iteration, chances of them heading towards the wrong direction were eliminated.

() **Root cause analysis:** This aspect was not discussed with participants from company Y.

(+) **Refactoring – Code refactoring:** The responses to the interviews indicated that refactoring was used to manage efficiency and also to reduce technical debt in this company. Refactoring was therefore indicated as having a positive influence on team productivity in this organisation. Obtaining feedback quickly was viewed as enhancing efficiency.

(-) **Self-organising:** The data-collection information indicated that the agile team was not self-organising. There was no direct communication between the developers and users (i.e. customer). Participant LP1 was quoted saying: '*so it's not a self-organising team, so communication cannot happen by itself, in fact, it's a self-disorganising mechanism (participant LP1)*'. Agile principles lay emphasis on agile teams being self-organising and in this organisation, self-organising was perceived as having a negative effect on team productivity.

(-) **Project constraints:** Project constraints was indicated by participants as having a negative effect on team productivity, because if there were constraints in the project, for instance, client continuously changing their mind on what they needed the final product to be, there were a lot of delays which led to projects not completed at all.

(+) **Work environment – Work setting:** In this company the working environment was considered as having a positive influence on their team productivity. The developers sit together in an open office next to each other and are able to share information. The data collection indicated that the developers were only working from their offices and not at client sites. Figure 4.5 shows the developers sitting arrangements.



Figure 4.6 *Developers work setting (Company Y)*

4.5.4 Data collection – Definitions and metrics of software productivity

Data collection in this section covers definitions of software productivity and metrics from the agile team perspective.

4.5.4.1 Company X

The data collection information in this company indicated that there was no universal definition for productivity. The definitions of productivity from different participants in this company included functionality, delivering working software that is of good quality and useable.

The following were some of the definitions of software productivity given by participants:

- *'Delivering functional software to your client, timeously, and something clients believe helps them in their process (participant TL1)'.*

- *'It is output, the functionality that gets produced, and must be of good quality and useable (participant TL2)'*

From the discussions, productivity was perceived by the participants as producing code that is of high quality and can be useable as opposed to producing a lot of code with no business value.

There were no formal metrics used to measure team productivity in Company X. However, they did measure team productivity by using story points that were completed in a sprint.

4.5.4.2 Company Y

The data gathered indicated that the company was using a specific formula to measure their team productivity.

The following were some of the software productivity definitions given by participants:

$$\text{'Software productivity'} = \frac{8 \text{ hours per day} - \text{hours per day spent on servicing technical debt}}{8 \text{ hours per day}}$$

The following was an example given to illustrate how the formula works:

'For example, if four hours in a day is spent on fixing bugs due to developers not following processes or using code that was easier to implement instead of applying the overall solution, software productivity will be as follows:

$$\text{Software productivity} = \frac{8 \frac{\text{hours}}{\text{day}} - 4 \text{ hours/day}}{8 \text{ hours/day}} \quad 8 \text{ hour day productivity} = 0,5$$

In other words, if half of our time is wasted, then the productivity level is 0,5, otherwise, if our technical debt takes the whole day, our productivity level is zero (participant LP1).'

Therefore, if developers in this company spent most of their time on extra development work that was caused by initially using code that was easy to implement instead of applying and following processes, the level of their technical debt increased to eight hours in a day,

productivity level became zero, and the programmers were not productive at all because they spent the day fixing bugs.

In company Y, metrics that they have were a subjective feel. The company used estimate accuracy as metrics to measure their team productivity although the responses from participants indicated that it was difficult to measure some variables such as technical debt because people were involved in developing the software and it was difficult to accurately estimate how much time developers actually spent on fixing bugs rather than producing the real code.

The following were comments quoted from participants:

- *'So we track for all of our teams that sum of the estimates for iterations vs how much the resource was in the iteration (participant LP1)'.*
- *'Part of that ratio is the technical debt. Part of the ratio is just simple estimating inaccurately because there is no way people know how much it's going to take in software because there is just too many known unknowns and unknown unknowns with a story (participant LP1)'.*
- *'So, using that ratio, we got information to plan. We have a rough idea of how long it will take - a certain amount of time to be completed. But that ratio also is a rough indicator of technical debt. If that ratio starts to get really poor, this indicates your technical debt is going up. But it's not an exact measure. There is no way to measure the amount of time you're spending on servicing technical debt (participant LP1)'.*

The evidence gathered indicated that although the company was using these metrics to measure productivity, it was still difficult to measure it in agile teams since software is built by people and clients' requirements change all the time.

4.5.5 Data collection – Agile productivity metrics and performance monitoring

Data collection in this section covers agile productivity metrics and performance monitoring in agile teams.

4.5.5.1 Company X

Company X does not use anything formal but does measure agile team productivity on a case-to-case basis. There were no performance monitoring mechanisms in place.

The following comments were made by participants during the interview discussions:

- ‘We don’t monitor, the only monitoring we do is, we use story points to determine how much work we need to do (participant TL2)’.

The only performance monitoring tool that was in place was the use of a burndown chart which showed how many story points have been done over time. An example of a burndown chart is shown in figure 4.7 below:

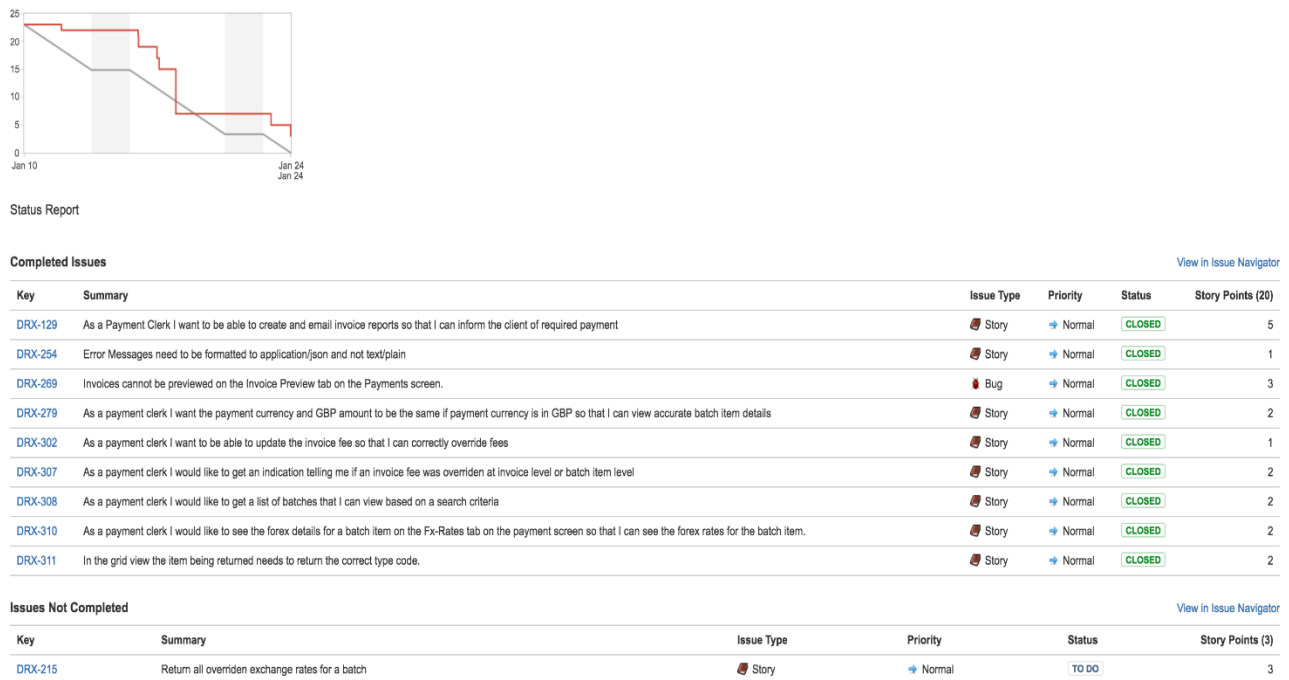


Figure 4.7 Burndown chart with summary

4.5.5.2 Company Y

Company Y data collection revealed that there were no separate metrics that could be used to measure their productivity. When measuring their agile team productivity, the use of regression testing, refactoring, continuous integration and deployment to make the feedback loop shorter was used as the source for measuring their productivity.

The following comments were noted from the interview discussions:

- *'There is no direct way to measure productivity (participant PM1).'*
- *'We can measure how well people are doing in those things and are going to give you feedback sooner and therefore cost you less to fix them (participant PM2).'*

For performance monitoring, company Y uses JIRA (software for tracking stories) to track their stories. The programmer gives an estimate of how long the piece of work will take to complete. The Product Owner loads this estimate in JIRA to track the story. The ratio of the estimated time to the actual time it took to complete the task was to be applied for future estimates.

The following comment was obtained from the interview discussion:

'So we keep our team's record of estimate accuracy and apply that ratio to all our future estimates, to be able to do iterations and release plans (participant PM1).'

For performance monitoring in this organisation, past work with almost similar specifications or amount of story points was used to give a rough indication on how much time was expected for each development team to complete a project.

4.5.6 Data collection – Monitoring productivity in self-managed teams

Data collection in this section covers monitoring productivity in self-managed teams.

4.5.6.1 Company X

In company X, the data collection revealed that nothing formal from management had been put in place to monitor productivity of each team. The tools that were in place were build-up charts and the use of story points.

Rescue time was the software that was mentioned in the discussions and was used to track time spent on applications running off each developer's machine and give a report to show

which applications were productive. Figure 4.7 and 4.8 show the reports of the activity that were productive in a week.

Activity <small>select multiple</small>	Category & Productivity Score		
parallels desktop	SOFTWARE DEV : GENERAL	<input checked="" type="radio"/> VERY PRODUCTIVE	DELETE
slack	COMM/SCHEDULE : INSTANT MESSAGE	<input type="radio"/> NEUTRAL	DELETE
microsoft remote desktop	SOFTWARE DEV : GENERAL	<input checked="" type="radio"/> VERY PRODUCTIVE	DELETE
youtube.com	ENTERTAINMENT : VIDEO	<input checked="" type="radio"/> VERY DISTRACTING	DELETE
52.50.3.9	SOFTWARE DEV : QUALITY ASSURANCE	<input checked="" type="radio"/> VERY PRODUCTIVE	DELETE
vat-it.atlassian.net	BUSINESS : ADMINISTRATION	<input checked="" type="radio"/> VERY PRODUCTIVE	DELETE
loginwindow	UTILITIES : GENERAL	<input checked="" type="radio"/> PRODUCTIVE	DELETE
microsoft outlook	COMM/SCHEDULE : EMAIL	<input type="radio"/> NEUTRAL	DELETE
d1a05f84415c34edb5b57181a52bf323.eu-west-1.aws.found.io	SOFTWARE DEV : QUALITY ASSURANCE	<input checked="" type="radio"/> VERY PRODUCTIVE	DELETE
google.co.za	REFERENCE : SEARCH	<input type="radio"/> NEUTRAL	DELETE
localhost:3000	SOFTWARE DEV : GENERAL	<input checked="" type="radio"/> VERY PRODUCTIVE	DELETE
whatsapp	COMM/SCHEDULE : INSTANT MESSAGE	<input checked="" type="radio"/> DISTRACTING	DELETE
superbalist.com	SHOPPING : CLOTHES & PERSONAL	<input checked="" type="radio"/> VERY DISTRACTING	DELETE

Figure 4.8 Activity setup rescue time



Figure 4.9 Rescue time weekly report

4.5.6.2 Company Y

The Company Y data collection revealed that there was no direct monitoring tool for self-managed teams. There was no way they could quantify speed independently of complexity and complexity couldn't be quantified based on the features. The following comment was noted from one participant:

- *'Time is a feature that looks very simple, like the undo button to the client is just a click, but is actually a complicated thing to do in the code (participant PM1).'*

The responses clearly indicated that it was difficult to directly monitor productivity in self-managed teams.

4.6 Data analysis – Productivity factors

This study involved gathering data from participants working in two different companies that are located in two geographical areas. The data gathered during the interviews is analysed and presented in this section.

The following are interpretations and notations used for data analysis

(+) Positive influence
(-) Negative influence
(+/-) Mixed influence
() Not discussed/ Not applicable

Table 4.4 summarises the effect of each factor on the team productivity in each company under study. The table shows that although the factors that affect each team productivity are more or less the same, each company's perceptions of the effect was different. As software development companies in South Africa strive to reduce time to market, to produce software at lower cost and to be competitive, this study may be of use to companies that are in the process of adopting agile methodologies.

Table 4.4. A summary of the productivity factors effect to companies under study

Productivity Factor	Coding	Company X	Company Y
Process	Following procedures	(+/-)	(+/-)
Knowledge	Team member skills	(-)	(+/-)
Team morale	Team motivation and commitment	(+/-)	(+/-)
Management style and quality	Management support	(+/-)	(+/-)
Team size	Number of team members per project	(+/-)	(+)
Team collaboration	Team cohesiveness	(-)	(+)
Daily meetings	Daily 5 -10 minutes stand-ups	(+)	(+)
Continuous integration	Response to change	(+)	(+)
Size of database	Size of features developed	()	(+)
Product complexity	Project/ product complexity	(+/-)	(-)
Software tools used	Programming tools	()	(+/-)
Employee turnover	Team member turnover	(-)	(+/-)
Multisite development	Geographical location	(+/-)	(+)
Code reusability	Code reuse	()	(+)
Required software schedule	Duration, Deadlines, Timelines	()	()
Real customer involvement	Onsite customer	(+/-)	(-)
Planning meetings	Weekly, bi-weekly meetings	(+)	(+)

Productivity Factor	Coding	Company X	Company Y
Root cause analysis	Formal and informal investigation	()	()
Refactoring	Code refactoring	(+)	(+)
Self-organising	Self-organisation	(+)	(-)
Project constraints	Resources, Schedule, System or Environment	(-)	(-)
Work environment	Work setting or sitting arrangement	(+/-)	(+)
Pair Programming	Two programmers working together	(-)	(-)
Positive effect		5	10
Negative effect		5	5
Mixed effect		8	6
Not discussed/ Not Applicable		5	2

4.7 Summary

This chapter presented the data-collection information from the two South African companies that adopted agile methodologies in their software development. Semi-structured interviews were used as the main data collection method in this study. The interview process was presented in detail, data analysis was discussed in detail and the foundation for data collection was also presented. A summary of the profiles of the companies under study was presented. The interview responses from participants were discussed in detail and data were analysed and coded.

The next chapter discusses the findings and the limitations of this study.

Chapter 5: Discussion of findings

5.1 Introduction

Chapter 4 presented a detailed discussion of the data-collection information from two South African companies that adopted agile methodologies in their software development. The data collection process, which consisted of semi-structured interviews and data analysis, was also discussed in detail. A detailed discussion of the participants' responses was also presented. This chapter will discuss the findings in detail.

5.2 The results

The objective of this study was to develop a framework to understand the factors that affect agile team productivity. Below is the description of the results obtained from case studies analysis, literature study and other evidence which support the key findings.

5.2.1 Productivity factors

Numerous factors that influence agile team productivity were identified in the literature study (Boehm *et al* 2000:15-39; Maxwell & Forselius 2000:80; Kitchenham & Mendes 2004:16 and Berntsson-Svensson & Aurum 2006:3) and in the case studies. The identified factors that influence agile team productivity in the case studies were more or less the same as the ones identified from the literature study.

For companies that adopted agile methodologies in their software creation, understanding the factors that influence their team's productivity is very important. Information obtained from the case studies indicated that a number of identified factors affected team productivity in both positive and negative ways. The first research finding on agile productivity was that the success of any project is highly dependent upon human efforts. Team morale, commitment and teamwork were identified as the most influential factors in agile team productivity in the case studies. Personal motivation and skills are seen as the drivers of any successful agile project. This

findings in the case studies relate to Trendowicz and Münich (2009), who found out in their research that software project success depends on human.

The other factors that affect team productivity are tools and processes. The tools that developers need to do their work and the processes that they need to follow influence productivity marginally in the companies under study. However, having the best tools and processes in place cannot alone be considered a substitute for motivated skilled employees and effective work coordination. Companies still need to invest in their employees to bring more benefits than only in tools and processes. Boehm *et al* (2000) also found out that software tools used are necessary but they are not very important as compared to experienced and motivated skilled employees.

Product complexity and work environment are other factors that influence agile team productivity. Planning at a higher level, getting the customer involved at early stages and determining the work scope correctly was found helpful for increasing the productivity levels. This findings is related to Berntsson-Svensson and Aurum (2006), where they found out that detailed project scope, complete and precise requirements, accurate schedule estimates, and involvement of the customer in the development process is significant in order to achieve high productivity. Although product complexity and work environment may have had a neutral effect on productivity, the study revealed that the factors that really influence productivity rely mainly on the humans involved in developing the software. Maxwell and Forselius (2000), in their research, found out that business domain is important as it can be a motivational factor to employees. Therefore, management should provide a conducive working environment, tools that are needed by developers to get the job done, and ensure team morale is always high.

5.2.1.1 Agile practices

Agile practices provided for by the Agile Alliance, if followed correctly, are very useful in agile team productivity as they provide teams with procedures they can follow to deliver a successful project. Most agile practices - such as acceptance testing, backlogs, continuous integration, refactoring, retrospective, scrum events, user story mapping and information radiators - were being applied in the companies under study and they proved to be helpful in

improving their team productivity. Agile 101 provides important Agile Manifesto guidelines which are key practices that support teams in implementing and executing with agility (Agile Alliance, 2017). One practice not being used was pair programming, because it was perceived as having a negative effect on team productivity. In the case studies, it was noted that differences in developers' personalities made it difficult for two developers to work on the same code at the same time, despite pair programming's advantage of continuous code review. Even though pair programming was perceived as having a negative effect on productivity, agile teams could probably use it for training purposes.

5.2.2 Importance of productivity

The adoption of agile methodologies by South African software development companies is due to the need to respond to change more quickly and to succeed in software development. According to the respondents in the case studies, agile methodologies have proved to be extremely important to companies that adopt them in South Africa because their productivity levels increased. Agile methods enable teams to do work that is important for the client, no time is wasted building software that is not needed, and it has proven to encourage a lot of engagement between the customer and the development team. Adoption of agile methods gave the companies under study a competitive advantage since they are now able to do work at lower cost than their competitors in the software industry. Therefore, by adopting agile methodologies, companies in South Africa reaped a lot of benefits which even increased their team productivity since the agile approach helped these companies to deliver a better and more relevant end product.

5.2.2.1 Definition of software productivity

Software productivity is simple to define in theory but difficult to define in a real working environment. There was no clear definition of software productivity provided in the case studies, nor in the literature study. The definition of productivity remains a mystery in agile software development, and no exhaustive definitions have been established yet (Zimmermann 2017:1 and Tangen 2002:18). Productivity in agile teams still remains a

contentious issue; there are just too many known unknowns and unknown unknowns in software development, which makes it difficult for management to measure productivity in agile teams accurately. The research findings indicated that productivity in agile teams goes beyond measuring the ratio between the outputs and inputs; human factors like morale, motivation and commitment play a significant role as a determining factor for measuring team productivity.

5.2.3 Metrics

The commonly used metric in agile teams is story points or completed user stories in a sprint; sometimes called velocity. Metrics in agile teams assist in gauging the team's productivity, predictability, health, and work quality. The metrics that were used by the companies under study in South Africa are visibility metrics, efficiency metrics, predictability metrics, story point estimation or estimate accuracy, wherever they track, for all their teams, the sum of the estimates vs how much resource was in the iteration. Productivity in these case studies was determined by using story points completed in a sprint, as displayed on their velocity charts. According to Ambler (2016), if management can easily measure productivity they can easily identify what is working for the team in given situations, or what is not working for the team, and adjust accordingly. He went on to point out that one way to do so is to look at acceleration, which is the change in velocity but also noted that it's generally impossible to use velocity as a measure of productivity. Two different teams velocity cannot be compared because they are measuring in different units (Ambler 2016).

Even though there were no formal metrics being applied in the companies under study, a number of agile metrics could be suitable for determining productivity in agile teams:

- process health metrics for assessing day-to-day delivery,
- release metrics which identify impediments to continuous delivery,
- product development metrics which help align product features to user needs,
- technical or code metrics which help determine quality and people, or

- team metrics which reveal issues influencing a team, and level of engagement, might be suitable metrics for determining productivity in agile teams.

Organisations with agile teams can benefit from using metrics so as to allow them to assess effort, performance and productivity of their teams over time.

5.2.3.1 Adjustments to promote productivity

The researcher's finding was that there were indirect adjustments that could be made to promote productivity in situations where agile metrics have a negative effect. Agile teams in the case studies could consider their sizing method because if productivity is affected in a negative way, it means that the stories are being sized wrongly or the developers are simply working on bugs (technical debt) and productivity is not showing. Developing trust across the agile team is equally an important adjustment to improve product quality, development speed through early release and productivity.

5.2.4 Productivity monitoring

Monitoring the productivity of an agile team involves tracking the team's progress to ensure that the team is effective and efficient. In the case studies, there were no monitoring or formal metrics in place to measure how well these companies' agile teams were doing. The monitoring that was done was on a case-by-case basis and informal. Story points were used by the companies under study to determine how much work their teams needed to do in each sprint.

However, although the finding indicated that there were no team productivity monitoring tools put in place in these companies under study, for the productivity of their teams to improve, formal monitoring tools or metrics need to be incorporated so as to assist management to assess the teams' progress on each project. Monitoring needs to be done not only when things go wrong but at all times, to improve productivity.

5.2.4.1 Agile metrics and productivity metrics

Metrics are used to acquire better insight into the work performed by teams and give a better understanding of the current situation of changes over a certain period of time (Pressman & Maxim 2015:724). Without metrics, reviewing any development or effort is likely to be open to bias and gut-feeling-based interpretation. Agile metrics are used to measure phases of the software development process whilst mainly focusing on software delivery. Productivity metrics are used to measure output produced over a specified period. Agile metrics relate to productivity metrics in that both metrics measure performance and they provide measurable goals for developers. From the case studies, it was noted that both metrics helped in building an agile team's culture. Agile metrics and productivity metrics should therefore be used as leading indicators for change management, affording an opportunity to review, assess, and analyse the cause in time.

5.3 Updated proposed conceptual framework for agile team productivity

Figure 5.1 shows the updated proposed conceptual framework for factors that influence productivity in agile teams.

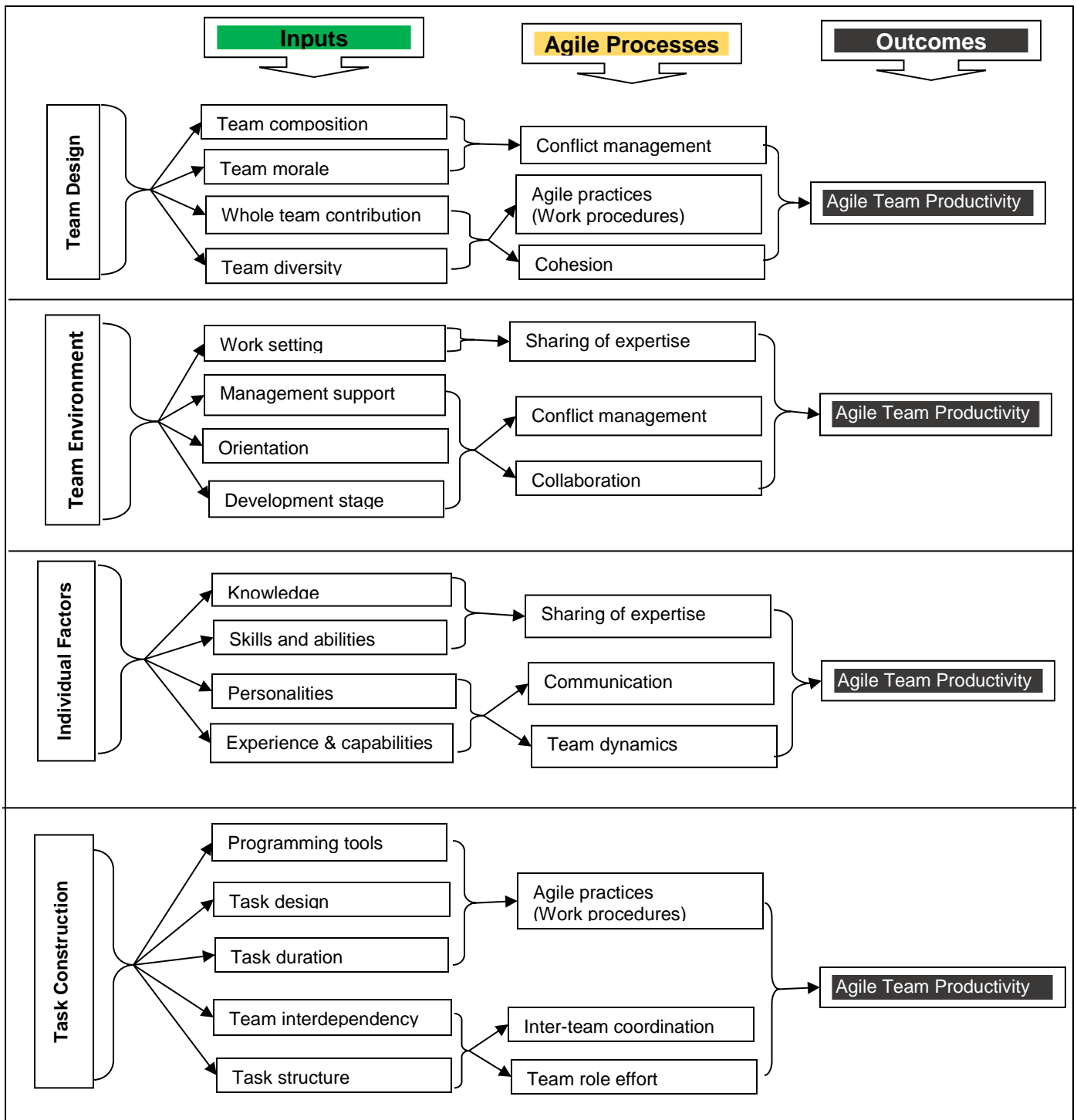


Figure 5.1 Updated proposed conceptual framework

5.4 Limitations of this research

Even though this research presented many contributions that could assist agile team productivity or companies that would like to adopt agile methodologies in their software development, a number of limitations were noted, too:

5.4.1 Unavailability of participants

During the data collection, the whole agile team working on the same project was not available to be interviewed at the same time during focus group interviews. In most cases only two developers with their team leader would be available, which meant the researcher was obtaining feedback from only a few participants rather than having the whole team in the discussion. Other participants who were scheduled to be interviewed were busy with their work or travelling elsewhere on the interview day, and the researcher was not able to conduct those interviews.

5.4.2 Lack of available data

In some cases, participants were not willing to answer certain questions which were important to this study. Although follow-up interview questions were sent to some participants by e-mail, for clarification purposes, participants were not willing to put things in writing and could only respond with some information and not all the facts. Also, access was denied to the researcher to most of the reports, for example the JIRA reports, for measuring team productivity, which could have assisted during data analysis.

5.4.3 The research strategy

A case study research strategy was used in the data collection. The case study research is also a limitation to this study as it doesn't allow the researcher to make generalisations about the findings.

5.4.4 Researcher's access

The researcher's access to people and information was limited to some extent. The companies under study were competitors in the agile software development industry and access to certain reports and information was limited due to the sensitivity of the data. This study was therefore limited to reliance on what participants had to say in the interviews, which had to be taken at face value, and further facts could not be obtained using these reports.

5.5 Possible future research

Although the findings in this study provide reliable information that software practitioners, management and other stakeholders could use to enhance productivity in their agile teams, there is still need for further empirical scrutiny on agile team productivity factors and monitoring approaches. Future research may consider mapping agile team productivity factors with agile team productivity monitoring approaches, to come up with metrics that could help improve productivity in agile teams.

5.6 Summary

Chapter five presented the research findings, the updated proposed conceptual framework for factors that influence agile team productivity, research limitations and possible future research. The research findings indicated that there is still much to learn about factors that affect agile team productivity and productivity monitoring approaches in agile teams. The finding that agile metrics relate to productivity metrics is somehow not conclusive. The researcher couldn't get access to the software (JIRA) that the companies under study were using to observe the trends in productivity level over time. Further research needs to be conducted using a much bigger sample.

Chapter 6: Research summary and conclusion

6.1 Introduction

Chapter 5 discussed the research findings for this study. An updated conceptual framework to understand factors that influence agile team productivity was proposed. Limitations of this research and possibilities for future research were also discussed. The aim of this chapter is to ascertain whether the initial research questions in this study have been answered and whether reliable solutions have been obtained using the qualitative research method.

6.2 Research summary

The effect of productivity factors in software engineering is often neglected by organisations engaged in software development. Productivity of software development teams is very important in organisations that adopt agile development methodologies in their projects. Companies require shorter time-to-market for their products whilst keeping their development cost low. Measuring productivity in agile teams has been noted as a better approach to assessing their performance and in order to ensure continuous software delivery. While there are issues surrounding productivity measurements, what is more relevant is whether the factors that affect agile team productivity are a drawback to successful delivery of agile projects. This research explored this issue by focusing on agile productivity factors, metrics and the benefits of adopting agile methods.

Productivity factors are acknowledged to have an effect on team productivity, yet exactly how remained to be proved. Identifying these factors and their level of importance not only makes it possible to apply the correct metrics in measuring agile team productivity but also to ensure that quality software is delivered frequently. The evidence drawn from the case studies indicated that it is also important to assess the work of agile teams and to apply the correct metrics when measuring agile team productivity. This research explored the factors that affect agile team productivity by using COCOMO II (Boehm *et al*, 2000). Considering the importance of productivity and emerging research in agile team productivity, the study was seen as important.

Productivity factors, definition of and metrics for productivity were the foundation for this study and the in-depth analysis of the level of effect of these factors on agile team productivity formed the basis for this research journey.

6.2.1 Overview of the chapters

Chapter one presented the introduction, problem statement, research questions, and research objectives. This chapter provided the foundation for understanding the problem statement.

Chapter two presented a literature review on factors that affect agile team productivity, most frequently claimed benefits of adopting agile methods, the concept of agility, definitions of productivity, and productivity metrics.

Chapter three discussed the research methodology, and the research method that applied to this study was selected. The research approach was discussed, an analysis of quantitative and qualitative research was presented. The research study followed a qualitative research methodology. Five qualitative research strategies were considered, grounded theory, ethnography, narrative research, phenomenological research, and case studies; case studies was the selected research strategy.

Chapter four presented the data collection from case studies. A discussion of the data collection process and the participants' responses was presented. The research involved data collection from two South African companies (one in Gauteng Province and the other in Western Cape Province) that had adopted agile methods in their software development for more than three years.

Chapter five discussed the findings. The key findings were discussed, limitations of the research were noted and possible future research was proposed.

Chapter six presents the research summary and conclusion. This chapter highlighted contribution to knowledge, and provided a summary of the achievements of this study.

Chapter four and five provided a detailed discussion of the outcome of the original research questions.

The research questions were identified as follows (see section 1.4):

RQ1. What are the factors influencing productivity of agile teams and how do these factors effect on team productivity from the team point of view?

SubRQ1.1. Which agile practices affect a team's productivity?

RQ2. What is the importance of productivity on companies that adopt agile methods?

SubRQ2.1. How would they define software productivity?

RQ3. What are suitable metrics for determining agile team productivity?

SubRQ3.1. In scenarios where such metrics have negative effects, are there adjustments that could be made to promote productivity?

RQ4. How should productivity factors in agile teams be monitored, considering agility and adaptability?

SubRQ4.1. How do agile metrics relate to productivity metrics?

6.3 Summary of objectives achieved

This section provides a discussion of the approach incorporated in this study focusing on the problem statement, research questions, objectives and research outcomes. A discussion which shows that the four research questions listed above were accurately investigated is presented below.

Question number one resulted in the identification of common factors that affect agile team productivity from both case studies and literature study. The research indicated that, even though numerous factors have an effect on team productivity, understanding which factors have the greatest effect on each team's productivity is important in applying corrective action to

increase productivity. The research showed that incorporating agile practices that suit each organisation's team structure, and working on factors that affect team productivity, could help organisations identify the root cause of the problem, which might result in lower development costs, time saving and frequent software delivery.

The research outcome for question number two indicated that adoption of agile methodologies in South African software development companies under study was important. Agile methodologies enable teams to focus on important items during the development process and to avoid unproductive activities. Even though in the case studies productivity was not defined clearly, the teams demonstrated that they do have an understanding of what being productive means.

The overall outcome of question number three was that story points were the commonly used metric in these case studies; they assist in measuring the agile team productivity. Using story points allowed the companies under study to track the progress of their teams by looking at the estimated time to complete a sprint, versus the actual time; this was used for future estimates as well. Although metrics were not being applied fully in these companies, incorporating them into day-to-day work could improve their productivity levels.

The final research outcome resulted in the development of a conceptual framework for agile team productivity.

Productivity factors in agile teams could be monitored by looking at the following:

- Team design: The team should have an adequate number of developers (ranging from three to seven, depending on the project size) and be encouraged to gel.
- Team environment: The work setting should be user-friendly for developers to work in.
- Individual factors: The developers' skills, knowledge, experience and personalities should be considered when forming the team.
- Task construction: The task duration should be estimated using reasonable or previous completion estimates, and the necessary tools should be available for developers.

Using the proposed agile team productivity conceptual framework, it was found that the framework could help the companies under study (or any other company that adopts agile methods) to improve the productivity of their team. The proposed conceptual framework could be used as a starting point to understand the factors that affect each agile team productivity and devise ways in which the productivity in these teams could be suitably monitored.

6.4 Summary of contributions

This research has contributed to the management of software development teams in companies that adopted agile methods to improve productivity. The benefits of adopting agile methods and the factors that influence productivity were analysed.

Understanding the factors that affect productivity in agile teams brings about significant insight into the way that they work, motivating teams to work together and applying the same metrics across the board when tracking each team's progress.

The evidence from this study has shown that creating a stable working environment, providing the necessary tools and tracking an agile team's work every day results in improving team productivity. The research has also proven that following agile practices has a positive influence on productivity.

It should be recognised that using metrics to measure productivity in agile teams is helpful, metrics assist in creating a team's culture and allow the team to gel.

Successful software delivery is only possible if people are committed to their work, are provided with the necessary tools and have access to a stable working environment. Individual factors like knowledge, skills, abilities, personalities and experience should be considered when forming agile teams. Consideration of these factors will result in groupings of people who are able to work together to achieve a common goal, which is an important component in improving productivity.

In this study, the following facts have been established:

- Understanding the factors that affect the productivity of any agile team is important to ensure frequent software delivery.
- Using metrics to track the productivity of an agile team can be helpful.
- Only agile practices that are perceived to have a positive effect on productivity should be used.
- Monitoring productivity in agile team is important to ensure that teams are effective and efficient.

6.5 Recommendations for implementation

Agile teams need both management support and customer engagement for successful project delivery. Besides, other important factors include:

- Agile team commitment, in listening and responding to problems quickly.
- Increasing team morale by giving developers challenging work.
- Recruiting team members with compatible personalities to work in the same team.
- Providing the relevant tools needed to execute projects.
- Providing a stable working environment.
- Incorporating formal metrics to ensure fairness in monitoring and measuring the productivity of each team.

6.6 Future research opportunities

Although this research report already strongly emphasises agile team productivity factors and metrics, there is still a need for further research on this topic. Based on the evidence and findings obtained from this research, more companies that adopted agile methodologies could be studied.

6.7 Conclusion

Software productivity still remains a challenge in the software industry to this day. Companies need to keep up with frequently delivering software in order to increase their productivity rate. The adoption of agile methodologies in software development by most companies is intended to help them lower development costs and deliver software within a shorter space of time.

In the literature, the importance of measuring productivity in any workplace has been recognised, but the actual measurement of productivity in agile teams is still under development, and there is a need for uniform metrics to ensure that the productivity of each team becomes visible. Factors that affect the productivity of an agile team are acknowledged as the most influential drawbacks to any successful software project delivery. This investigation has provided a list of negative and positive factors that affect agile team productivity; the metrics that might be suitable to measure productivity in agile teams; benefits that companies may reap from measuring and monitoring productivity in self-managed teams; and a proposed conceptual framework to manage issues surrounding agile team productivity. In this regard, further examination is needed with a bigger sample to establish whether monitoring and measuring agile team performance may result in increased productivity.

APPENDIX A - INTERVIEW PROTOCOL

INFORMATION SHEET

Introduction

Viola Nzou (student number: 46563946) is a Master of Science in Computing student from the University of South Africa (UNISA) and is conducting a study to evaluate factors that affect productivity of software development in agile teams, in order to come up with ways to improve the quality and time spent to deliver software in organisations. The researcher is interested in learning about the delivery of software to customers and how efforts to improve software productivity are working in practice from the viewpoint of software practitioners, customers, developers, and other stakeholders.

What are the reasons for carrying out this research?

The researcher would like to know more about how software development productivity is being handled in your organisation. To do this, we are asking selected software practitioners, customers, developers, and other stakeholders' questions about their viewpoint on software delivery and effects on productivity. This information will assist us to understand how and why software productivity in agile teams is not improving in the organisation.

What happens if I participate in this research?

The researcher would like to ask you some questions about your viewpoint on software development productivity, its effects on agile teams and how it is measured. The researcher will be taking notes of the interview, and a recording using a digital voice recorder will be made. After the researcher has interviewed you, you will not be asked to do anything further. All information and data gathered will be treated with confidentiality by the researcher, and interviews records will be kept securely in locked cabinets. Personal identification information, for example names, will not be used in any reports arising out of this research.

What is the duration of research?

The interview lasts about 60-90 minutes, but the total duration of the study will be about one year.

Can I stop participating in the research?

Yes, you can choose to discontinue participating at any given time. You just have to inform the researcher right away if you wish to discontinue the interview.

Are there any risks can be expected from participating in the research?

Taking part in any research study might involve a loss of privacy. Information provided about the participant's opinions and experiences shall be audio recorded, but no names will be used in any reports. No quotes or other results arising from your participation will be included in any reports, even anonymously, without your consent. The information gathered from these interviews will be used by the researcher only for her Masters dissertation and will be submitted to the University and placed in a secure place. The researcher will do her best to ensure that the personal information collected for this study is kept private.

What are the benefits of participating in this research?

There are no direct benefits to you from taking part in this research. Nevertheless, the information gathered will help the researcher and your organisation's management to understand how best to improve the productivity of software development teams.

Do I have any other options if I don't participate in this research?

It's your choice to choose not to participate in this research. If your decision is not to participate this research, there will be no penalty to you.

What are the costs of participating in the research? Are there any incentives for participating in this research?

There are no costs for participating in this research. No payment will be made for participating in this research.

What are my rights if I choose to participate in this case study?

Participating in this research is your own choice. It's your choice to either participate or not to participate in the research. If your decision is to participate in this research, you are allowed to change your mind at any point. Whatever your decision is, there won't be any penalty imposed on you in any way.

Who are the people I should contact for answers if I have any questions about the research?

Feel free to talk to the researcher about any concerns or questions you may have about this case study. Contact Viola Nzou on 0790878333 or her supervisor Prof E Mnkandla on telephone number 011 670 9059. If you have any comments, questions, or concerns about participating in this research, communicate with the researcher first. If for any reason, you don't wish to do this, or you still have concerns about doing so, you may contact the UNISA Ethical Committee at telephone number +27 12 429 3111.

Giving consent to participate in the research

You are welcome to retain this form if you like. Participation in this research is voluntary. You have the right not to take part in this research, or to withdraw from the study at any time without penalty. In case you don't wish to take part in this research, kindly tell the researcher immediately. If you do wish to take part in this research, please inform the researcher immediately. If you disagree to quotes and/or any other results arising from you taking part in this research being included, even anonymously, kindly inform the researcher immediately.

APPENDIX B - PARTICIPANT CONSENT FORM

CONSENT FORM

Study Title: Development of a framework to understand factors that influence software productivity in agile teams

Researcher: Viola Nzou 46563946 (MSc Computing, University of South Africa)

- The research study has been explained to me in a language that I understand. All the questions about the research have been answered. I comprehend what will happen during the interview and what is expected of me.
- I have been informed that I have a right to refuse to participate in the interview and that if I decide not to participate I don't have to give a reason, and that it won't prejudice the care that I can expect to receive now, or in the future.
- I have been informed that all my responses during the interview will remain completely confidential: my name and any other information that bears any of my identification will not be used.
- An explanation was given that sometimes the researcher might find it helpful to use my own words when reporting on the research findings. I understand that any use of my words shall be used completely anonymously (without my name). I have been informed that I can decide whether I authorise my words to be used in this way.

Circle response:

I agree to take part in the research:	Yes	No
I agree that my own words may be used anonymously in the report	Yes	No

Signature of participant:

PRINT NAME	SIGNATURE	DATE (YYYY/MM/DD)

Signature of researcher taking consent:

I have discussed the research with the participant named above, in a language he/she can understand.

I trust he/she has understood my explanation and agrees to participate in the interview.

PRINT NAME	SIGNATURE	DATE (YYYY/MM/DD)

APPENDIX C - INTERVIEW QUESTIONS

Individual / Group questions

- Q1. (a) Can you please give me a brief description of any current or previous software development project that you were involved in? What is or was the duration of the project? What methodology was used? Was the project delivered on time and did it meet the client's requirements and specifications?
- (b) What are the benefits of using agile methodologies?
- (c) How would you define software development productivity?
- (d) What are the factors that influence your team productivity?
- (e) Do these factors influence team productivity in a positive or negative way? Please elaborate further on your choice.
- (f) Can you please explain which agile practices have the most effect on your team's productivity?
- (g) How important are agile methodologies in your organisation? Are there any direct benefits from adopting agile methods?

Team design questions

- Q2. (a) Briefly describe your team composition, structure and diversity?
- (b) Do you think the management style in this organisation allows employees to be creative and innovative?
- (c) Does the leadership in this organisation motivate the team?
- (d) Looking at software development projects, are the problems encountered raised and/or forwarded to management in time?
- (e) When there are problems in developing the software, how are they managed? Who takes responsibility? Do all team members take responsibility for major effects?
- (f) Does your work setting affect your final product?
- (g) Do you know what the productivity rate in this company is? Please can you describe productivity along the lines of final product delivery?

(h) What is the engagement level between the team to the client during software development, since agile methods lay emphasis on developer, client communication and collaboration?

Individual and team factors questions

- Q3. (a) Agile software development is centred on responding to change. Is your team able to respond and handle change timeously?
- (b) How do you manage risks? Any there any examples that you can give me.
- (c) How do management measure team and individual productivity in this organisation? Are there any established metrics?
- (c1) Are there adjustments to promote productivity in situations where these metrics have negative effects? Do you think agile metrics relate to productivity metrics?
- (d) Which metrics do you think are suitable for determining productivity in agile teams?
- (e) Do you see members of the team working as a group or as individuals?
- (e) How do you manage “sudden change in requirements” during software development?
- (f)How do you monitor productivity of a self-managed team? Are there any processes put in place to monitor productivity, considering agility and adaptability?

Time management questions

- Q4. (a) How do you usually manage your time? Do you use any tool?
- (b) What time do you normally take breaks and for how long?
- (c) How do you balance your work and your personal life activities?
- (d) How do you rate your individual turnover?
- (e) How do you rate your skills and abilities?
- (f) How do you manage the duration of each project? How do you ensure that you deliver the final product on time?

Communication strategy questions

- Q5. (a) How best do you think team members, practitioners and customers communicate and collaborate among themselves?
- (b) In terms of deadlines, how well do you think each team member manages to meet deadlines?
- (c) Is there collaboration and transparency in this organisation?
- (d) Are your clients involved in software development process voluntary or are they being forced to be involved? Do you conduct a survey to customers and ask them if the features are actually useful before releasing the software?
- (e) How often do you have meetings as a group? Are these meetings beneficial to you and the team?
- (f) How do you ensure that team members are efficient?

APPENDIX D - PAST PAPERS PUBLISHED AND PRESENTED

1. A framework for enhancing productivity in agile teams: A South African study, *submitted to the African Journal of Information Systems (AJSI), October 2017.*
2. Understanding the factors influencing software productivity in agile teams. A case study of South African companies adopting agile methods. *Presented at the University of South Africa, School of Computing, Post Graduate Symposium, Johannesburg, South Africa, September 22, 2016.*

APPENDIX E - ETHICS CLEARANCE CERTIFICATE



Dear Ms. Viola Nzou (45563946)

Date: 2016-02-08

Application number:
009/VN/2016/CSET_SOC

REQUEST FOR ETHICAL CLEARANCE: (Development of a framework for understanding the factors influencing software productivity in agile teams)

The College of Science, Engineering and Technology's (CSET) Research and Ethics Committee has considered the relevant parts of the studies relating to the abovementioned research project and research methodology and is pleased to inform you that ethical clearance is granted for your research study as set out in your proposal and application for ethical clearance.

Therefore, involved parties may also consider ethics approval as granted. However, the permission granted must not be misconstrued as constituting an instruction from the CSET Executive or the CSET CRIC that sampled interviewees (if applicable) are compelled to take part in the research project. All interviewees retain their individual right to decide whether to participate or not.

We trust that the research will be undertaken in a manner that is respectful of the rights and integrity of those who volunteer to participate, as stipulated in the UNISA Research Ethics policy. The policy can be found at the following URL:

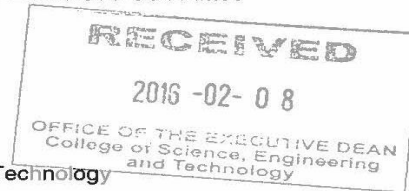
http://cm.unisa.ac.za/contents/departments/res_policies/docs/ResearchEthicsPolicy_apprvCounc_21Sept07.pdf

Please note that the ethical clearance is granted for the duration of this project and if you subsequently do a follow-up study that requires the use of a different research instrument, you will have to submit an addendum to this application, explaining the purpose of the follow-up study and attach the new instrument along with a comprehensive information document and consent form.

Yours sincerely

Prof Ernest Mnkandla
Chair: College of Science, Engineering and Technology Ethics Sub-Committee

Prof IOG Moché
Executive Dean: College of Science, Engineering and Technology



University of South Africa
College of Science, Engineering and Technology
The Science Campus
C/o Christiaan de Wet Road and Pioneer Avenue,
Florida Park, Roodepoort
Private Bag X6, Florida, 1710
www.unisa.ac.za/cset



APPENDIX F - LANGUAGE EDITOR CERTIFICATE

To whom it may concern

This is to certify that I, Klaus-Peter Klein, translator and language practitioner by profession, have edited a M.Sc. dissertation for Mrs. Viola Nzou.

The title of the thesis, containing approximately 40615 words, was

Development of a framework to understand the factors that influence software productivity in agile teams

Klaus P. Klein

805, 26th Avenue

Rietfontein, Pretoria

Cell 072-140 5300 / 072-493 6942

E-mail: klausklein@taalring.com

REFERENCES

- Abdel-Hamid, T. 1996. The slippery path to productivity improvement. *Software*, IEEE 13(4), p. 43 – 52.
- Abrahamsson P. (2005). Project Manager's Greetings - Agile Greetings, *Agile Newsletter Issue: 1*, 2004, p. 1.
- Abrahamsson, P., 2002. *The role of commitment in software process improvement*. PhD thesis, University of Oulu, Department of Information Processing Science & InforTech, Finland.
- Abrahamsson, P., Conboy, K., & Wang, X., 2009. 'Lots done, more to do': The current state of agile system development research. *European Journal of IS*, 18, 281 – 284.
- Agile Alliance, available at <https://www.agilealliance.org/agile-alliance-technical-conference-2016>
- Albrecht, A. J. 1979. "Measuring Application Development Productivity. In *Proceedings of the IBM Applications Development Symposium*. GUIDE/SHARE (Monterey, CA, Oct. 14-17). IBM, pp. 83-92.
- Allen, B., & Reser, D. 1990. Content analysis in library and information science research. *Library and Information Science Research* 12(3), 251 -266.
- Alliance, A, 2001. Agile manifesto. Online at <http://www.agilemanifesto.org>, 6(6.1) .
- Ambler, S.W., and Jeffries. 2002. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. New York. Wiley.
- Angkasaputra, N. & Pfahl, D. 2005. "Towards an agile development method of software process simulation," in *Proceedings of 6th International Workshop on Software Process Simulation Modelling*, p83 – 92, ProSim 2005, St Louis, Missouri.
- Arnold, M., and Pedross, P. 1998. *Software size measurement and productivity rating in a large-scale software development department*. Paper presented at the Proceedings of the 20th International Conference on Software Engineering, pp 490 -493.

- Avison, D. & Fitzgerald, G., 2006. *Information Systems Development*, 4th Edition. McGraw-Hill Education.
- Awad, M., A., 2005. *A comparison between agile and traditional software development methodologies*. University of Western Australia.
- Baskerville, R., and Dulipovici, A., 2006. The theoretical foundations of knowledge management. *Knowledge Management Research & Practice* 4(2), 83–105.
- Beck, K. 1999. *Extreme Programming Explained: Embrace Change*. Addison-Wesley. ISBN 978-0321278654.
- Beck, K. 2000. *Planning Extreme Programming*. With Martin Fowler. Addison-Wesley. ISBN 978-0201710915.
- Beck, K. et al. (2001). *Agile Manifesto* available at <http://agilemanifesto.org/>
- Beck, K., Andres, C., 2004. *Extreme Programming Explained: Embrace Change* (2nd Edition). Addison-Wesley Professional.
- Berntsson-Svensson, R. & Aurum, A. 2006. Successful software project and products: An empirical investigation. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical Software Engineering*, pp 144 – 153.
- Blackburn *et al.* 2000. Blackburn, J., Schudder, G., and Van Wassenhove, L. Concurrent software development. *Communications of ACM*, 43: 200 – 214, November 2000. ISSN 0001-0782.
- Boehm and Turner, 2006. Some Future Trends and Implications for Systems and Software Engineering Processes. *Systems Engineering*, Vol. 9, No. 1, Wiley Periodicals, Inc.
- Boehm et al. 2000. *Software Cost Estimation with COCOMO II*, Prentice Hall, Englewood Cliffs, N.J.
- Boehm, B. 1981. *Software Engineering Economics*, Prentice Hall, Englewood Cliffs, New Jersey.

Boehm, B., & Turner, R., 2004. *Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods*. Paper presented at the Software Engineering. Doi:10.1109/ICSE.2014.1317503.

Boehm, B.W, 1987. "Improving software productivity," *IEEE Computer*, September, 43 -57.

Bosch-Sijtsema, P.M, Ruohomäki, V., and Vartiainen, M., 2009. Knowledge work productivity in distributed teams. *Journal of Knowledge Management* 13(6), 533 – 546. Emerald Group Publishing.

Briand, L., Daly, J., and Wuest, J. 1998. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering: An Internal Journal*, 3, 65 – 117.

Brooks, W.D. 1981. Software technology payoff: Some statistical evidence. *Journal of Systems and Software*, 2(1), 3 – 9.

Bullock, J., Weinberg, G., & Benesh, M. 2001. Roundtable on project management: a SHAPE forum dialogue. Dorset House Publishing, New York.

Chan, F., and Thong, J., 2009. Acceptance of agile methodologies: a critical review and conceptual framework. *Decision Support Systems* 46 (2009), pp 803 – 814.

Checkland & Holwell. 1998. *Information, systems, and Information systems: Making sense of the field*. Chichester: Wiley.

Chemuturi, M., 2009. *Software Estimation Best Practices, Tools and Techniques: A complete Guide for Software Project Estimators*. J. Ross Publishing.

Clandinin, D. J. & Connelly, F.M. 2000. *Narrative inquiry: Experience and story in qualitative research*. San Francisco: Jossey-Bass.

Cobb, C., 2011. *Making Sense of Agile Project Management: Balancing Control and Agility*. Wiley Online Library.

Cockburn, A. 2001. *Agile Software Development: The Cooperative Game*. 2nd Edition. Pearson Education Inc. ISBN 10:0-321-48275-1.

Cockburn, A. 2001. *Writing Effective Use-Cases*, Addison-Wesley.

Cockburn, A. and Highsmith, J. 2001. *Agile Software Development: The People Factor*. IEEE Computer, vol 34 (2001).

Cockburn, A., 2002. *Agile Software Development*. Addison-Wesley.

Collier, K., 2001. *Agile Analytics: A value-driven approach to business intelligence and data warehousing*. Addison-Wesley.

Conboy, K. & Fitzgerald, B. 2004. Toward a conceptual framework of agile methods: a study of agility in different disciplines. In: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research, Newport Beach, pp 37 – 44.

Concas G, Damiani E, Scotto M and Succi G. 2007. *Agile Processes in Software Engineering and Extreme Programming*, LNCS 4536 pp 54 – 61. Springer-Verlay Berlin Heidelberg.

Creswell, J.W. 2003. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (2nd ed). Sage Publications, Thousand Oaks, USA.

Creswell, J.W. 2005. *Educational Research Planning, Conducting and Evaluating Quantitative and Qualitative Research* (2ed). Person Education, Upper Saddle River, USA.

Creswell, J.W. 2008, *Research design: qualitative, quantitative, and mixed methods approaches*, 2nd edition. Sage Publications.

Creswell, J.W. 2009. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (3rded). Sage Publications, Thousand Oaks, USA.

Crotty, M. 1998. *The foundations of social research*. London: Sage.

Dalcher. 2006. *Software Process: Improvement and Practice*. John Wiley & Sons, Ltd. Vol 11 Issue 2, p. 93.

Dale, C. J., and van der Zee, H. 1992. Software productivity metrics: who needs them? *Information and Systems Technology*, 34 No. 11, Nov 1992, 731 – 738.

Davis, A.M., Dietse, O., Hickey, A., Justisto, N., & Moreno, A.M. 2006, "Effectiveness of requirements elicitation techniques: empirical results derived from a systematic review",

- Proceedings of the 14th IEEE International Conference on Requirements Engineering (RE 2006)*, pp. 176 – 185.
- Davis, G.B., Ein-Dor, P., King, W.R., & Torkzaden, R. 2006. IT offshoring: History, prospects, challenges. *Journal of the AIS*, 7(11), 770 – 795.
- De Marco & Lister 1999. *Peopleware: Productive Projects & Teams*, Second Edition. New York. Dorset House.
- Deetz, S. 'Critical theory and postmodernist approaches to organisational studies', in S. Clegg. C.
- DeMarco, T. & Lister, T. 1987. *Peopleware: Productive Projects and Teams*. New York. Dorset House.
- DeMarco, T., & Boehm, B., 2002. The agile methods fray. *Computer*, 35(6), 90 – 92.
- Denzin, N. K., & Lincoln, Y. S. 2000. *Handbook of qualitative research*. Thousand Oaks, Calif, Sage Publications.
- Denzin, N. K.; & Lincoln, Y. S., eds. 2005. *The Sage Handbook of Qualitative Research (3rd ed.)*. Thousand Oaks, CA: Sage. ISBN 0-7619-2757-3.
- Depoy, E. & Gitlin, L.N. 2015. *Introduction to Research: Understanding and Applying Multiple Strategies*, 5th Edition. Elsevier Inc.
- Dey, I. 1993. *Qualitative data analysis. A user-friendly guide for social scientists*. London and New York: Routledge.
- Dilawar, A. 2011. "An Improved, Efficient and Cost Effective Software Inspection Meeting Process. Department of Software Engineering." University of Engineering & Technology, Taxila.
- Domegan, C & Fleming, D. 2007. *Marketing Research in Ireland: Theory & Practice*. Dublin: Gill & MacMillan, Limited.
- Drucker, P. F., 1999. Knowledge-Worker Productivity. The Biggest Challenge. *California Management Review*, 4(2) 79 – 95.

- Dybá, T., & Dingsoyr, T., 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50 (9-10): 833-859. ISSN 0950-5849.
- Eickelmann, N. 2001. *Defining Meaningful IT Productivity with a Balanced Scorecard*. Motorola Labs, Schaumburg, USA: p 67 – 70.
- Erl Thomas 2005. *Service-Oriented Architecture: Concepts, Technology and Design*. Prentice Hall, ISBN 0-13-185858-0.
- Garud, R., & Kumaraswamy, A., 2005. Vicious and virtuous circles in management knowledge: The case of Infosys Technologies. *MIS Quarterly*, 29(1), 9 – 33
- Gillham, B. 2000. *The Research Interview*. London and New York, Continuum.
- Gillham, B. 2000a. *Developing a Questionnaire*. London: Continuum.
- Guba, E J. 1981. Criteria for assessing the trustworthiness of naturalistic inquiries. *Educational Communication and Technology: A Journal of Theory, Research and Development*, 29(2), 75 – 91.
- Handbook of Research of Software Engineering & Productivity Technologies: Implications of Globalisation* (pp 28 – 37).
- Hardy and W. Nord (eds), *Handbook of Organisation Studies*. London: Sage.
- Highsmith, J., 2000. Retiring Lifecycle Dinosaurs, *Software Testing and Quality Engineering* July/August pp 22 – 28.
- Highsmith, J., 2001. “The great methodologies debate: Part 1.” *Cutter IT Journal* 14(12)
- Highsmith, J., 2002. What is agile software development? *CrossTalk – The Journal of Defense Software Engineering*, 1:4-9, October 2002.
- Highsmith, J., 2009. *Agile project management. Creating innovative products*. Upper Saddle River, NJ: Pearson Education.
- Highsmith, J., 2010. *Agile Project Management: Creating Innovative Products*. Addison-Wesley.

- Holden, M.T. & Lynch, P. 2004. Choosing the Appropriate Methodology: Understanding Research Philosophy (RIKON Group). *The Marketing Review*, 4 p 397 – 409.
- Hsieh, H, F., & Shannon, S. E. 2005. Three approaches to qualitative content analysis. *Qualitative Health Research*, 15(9), 1277 -1288.
- Iansiti, M. & MacCormack, A., 1997. Developing Products on Internet Time. *Harvard Business Review*, 75(5), 108 – 117.
- IBM Smart Cloud Orchestrator. <http://www.ibm.com>
- Jacobson, I. 2002. A Resounding “Yes” to Agile Processes – But Also to More, *Cutter IT Journal*, 15, 1. Pp 18 -24.
- Jalote, P., Palit, A., Kurien, P., & Peethamber, V., 2004. “Timeboxing: A process model for iterative development.” *Journal of Systems and Software*, 70 (2004): 117 -127.
- Jensen, R, 2014. *Improving Software Development Productivity: Effective Leadership and Quantitative Methods in Software Management*. Prentice Hall.
- Jones, C. 1986. *Programming Productivity*. McGraw-Hill, New York.
- Jones, C. 1987. *A short history of FP and feature points*. Internal report. Software Productivity Research, Inc., Burlington, MA: SPR Publication.
- Jones, C. 1996. *Applied Software Measurement: Assuring Productivity and Quality*. 2 ed. McGraw-Hill.
- Jones, C. 2008. *Applied Software Measurement*, 3rd edition, McGraw Hill.
- Karen A. Frenkel 1985, Toward Automating the Software Development Cycle, *Communications of the ACM*, Vol. 28, No.6 (June 1985) pages 578 – 590.
- Karlström and Runeson 2006. Integrating agile software development into stage-managed product development. *Empirical Software Engineering*, 11: 203 – 255, June 2006. ISSN 1382-3256.

- Kauffman, Banker, and Kumar. 1991."An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment." Unpublished.
- Kettunen, P., 2009. *Agile software development in large-scale new product development organisation: team level perspective*. Doctoral dissertation, Helsinki University of Technology.
- Kitchenham B, Al-Khilidar H, Ali Babar M, Berry M, Cox K, Keung J, Kurniawati F, Staples M, Zhang H, Zhu L. 2008. Evaluating guidelines for reporting empirical software engineering studies. *Empir Softw Eng* (3 (1)):97-121 doi: 1.1007/s 10664-007-90535.
- Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. 2009, "Systematic literature reviews in software engineering – a systematic literature review", *Information and Software Technology* 51, pp. 7 – 15.
- Kitchenham, B.A. & Charters, S. 2007. Guidelines for Performing Systematic. Literature Reviews in *Software Engineering Technical Report EBSE-2007-01*.
- Kitchenham, B.A., & Mendes, E. 2004). "Software Productivity Measurement Using Multiple Size Measures," *IEEE Transactions on Software Engineering*, 30(12): 1023 -1035, December 2004.
- Kumar, R. 2005. *Research Methodology: A step-by-step guide for beginners*. (2nd edition), London: Sage Publications.
- Larman, C., 2002. *Applying UML and Patterns: An Introduction to object-oriented analysis and design and the unified process*. 2nd Edition. Prentice Hall Inc.
- Lee, A. 2004. Thinking about Social Theory and Philosophy for Information Systems. In Mingers & L. Willcocks (Eds), *Social Theory and Philosophy for Information Systems*. Chichester Wiley.
- Lee, S., and Schmidt, R.C., 1997. Improving application development productivity in Hong Kong. In B.J.M & M.B.M (Eds). *Information technology and challenge for Hong Kong*. Hong Kong: Hong Kong University Press.
- Leedy, P.D., & Ormond, J.E. 2005, *Practical research: planning and design*. Prentice Hall, Upper Saddle River, N.J., 8th edition.

. MacCormack, A., Verganti, R., & Iansiti, M. 2001. Developing products on “internet time”: The anatomy of a flexible development process. *Management Science*, 47: 133 – 150, January 2001. ISSN 0025-1909.

MacCormack, A., Kemerer, C. F., Cusumano, M., and Crandall, B., 2003. Trade-offs between Productivity and Quality in Selecting Software Development Practices. *IEEE Software*, 20(5), 78 – 85.

Maskell, B. Associates Inc. 1999. *The Journey to Agility*, www.maskell.com/4box.htm

Maxwell, K., & Forselius, P. 2000. “Benchmarking Software Development Productivity,” *IEEE Software*, Vol 17, no 1, pp. 80 – 88.

McMillan, J. H. & Schumacher, S. 1993. *Research in education: A conceptual understanding*. New York: HarperCollins.

McMillan, J. H., & Schumacher, S. 2001. *Research in education: a conceptual introduction*. New York, Longman.

Mellor, S., 2005. Adapting agile approaches to your project needs. *IEEE Software*, 22(3), 17 – 34, May/June.

Melo, C., Cruzes, O.S., Kon, F., and Conradi, R., 2011. Agile team perceptions on productivity factors. In *Proceedings of the Agile 2011 (AGILE'11)*, pp 57 – 66, Salt Lake City, UT, USA, IEEE Computer Society.

Merriam, S. B. 2009. *Qualitative research: A guide to design and implementation*. San Francisco, CA: Jossey-Bass.

Miles, M. B., & Huberman, A. M. 1994. *Qualitative data analysis: An expanded source book*. Thousand Oaks, CA: SAGE Publications.

Mills, H.D. 1983. *Software Productivity*. Little, Brown & Co.

Mnkandla, E. 2010. “Agile Software Engineering” in Ramachandran, M, & de Carvalho, R. (Eds).

Mouton, J. 1996. *Understanding social research*. Pretoria. Van Schaik Publishers.

- Munhall, P.L. 1988. Ethical considerations in qualitative research. *Western Journal of Nursing Research*; 10:150 – 162.
- Myers, M. D. 2009. *Qualitative Research in Business & Management*. Sage, London.
- Myers, M.D. 2013. *Qualitative Research in Business & Management*. Sage Publications, London. Second edition.
- MyNatt, B.T. 1990. *Software engineering with student project guidance*. Englewood Cliffs, NJ: Prentice Hall.
- Noruwana, N., and Tanner, M. 2012. *Understanding the structure process followed by organisations prior to engaging in agile processes. A South Africa perspective*. SACJ No. 48, June 2012.
- Nogueira, J., Jones, C., and Luqi, 2000. "Surfing the Edge of Chaos: Applications to Software Engineering," *Command and Control Research and Technology Symposium*, Naval Post Graduate School, Monterey, CA, June 2000.
- Oates, B, 2005. *Researching Information Systems and Computing*. SAGE Publications Ltd.
- Oz, E., 2008. *Management Information Systems, 6th Edition*. Pennsylvania State University, Great Valley.
- Palmer, N. 2014. *Empowering Knowledge Workers*. Future Strategies Inc. ISBN 978-0-984976478. "Where is ACM Today?"
- Palmer, S. R., & Felsing, J. M. 2002. *A Practical Guide to Feature-Driven Development*. Prentice-Hall.
- Petersen, K. 2011. "Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology*, 53(4):317-343.
- Philip, J. 1998. *Project management*. London: ABC Publications.

- Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13, 303 - 337.
- Poel, K. & Schach, S. 1983. A software metric for cost estimation & efficiency measurement in data processing system development. *Journal of Systems & Software*, 3, p187 -191.
- Pressman R.S & Maxim B.R, 2015. *Software Engineering - a Practitioners' Approach. Eighth Edition*. McGraw-Hill Education.
- Pries-Heje, J., & Commisso, TH. 2010. Improving Team Performance. *Proceedings / Information Systems Research In Scandinavia (IRIS)*.
- Rajasekar, S., Philominathanet, P., & Chinnathambi, V. 2013. Research methodology. *Physics ed-ph*, 14. 1-53.
- Ramirez, Y. & Nembhard, D. 2004. Measuring Knowledge Productivity: A Taxonomy. *Journal of Intellectual Property*, 5 (4), 602 – 628.
- Rasch and Tossi. 1992. "Factors Affecting Software Developers' Performance: An Integrated Approach," *MIS Quarterly*. Vol 16, Issue 2, pp. 395 – 413, September 1992.
- Rasch, R.H. 1991. An investigation of factors that impact behavioural outcomes of software engineers. In *Proc 191 Conference on SIGCPR*, p. 38-53. ACM Press.
- Ravitch, S.M & Riggan, M. 2012. *Reason and rigour: how conceptual framework guide research*. Sage. Los Angeles.
- Riddle and Fairley 2012. *Software Development Tools*. Springer Science & Business Media, 2012.
- Robson, C. 2002, *Real world research: a resource for social scientists and practitioners researchers*, Oxford: Blackwell.
- Ross, A. & Ernstberger, K. 2006. Benchmarking the IT productivity paradox: Recent evidence from the manufacturing sector. *Mathematical & Computing Modelling*, 44(1-2), 30 – 42.

- Runeson, P., Höst, M. 2009. *Guidelines for conducting and reporting case study research in software engineering*.
- Saunders, MNK, Lewis, P, Thornhill, A and Bristow, A. 2015. 'Understanding Research Philosophies and Approaches' in MNK Saunders, P Lewis and A Thornhill, *Research Methods for Business Students, 7th ed.*, Harlow: Pearson Education. Chapter 4.
- Scacchi, W., 1994. Understanding Software Productivity. In Hurley, W.D. (Eds), *Software Engineering and Knowledge Engineering: Trends for the Next Decade* (Vol. 3, pp293 – 321), Piitsburgh, PA.
- Scacchi, W. 1995. Understanding Software Productivity: *Advances in Software Engineering and Knowledge Engineering*, D. Hurley (ed), Vol 4 pp 37 -70.
- Schach, S.R. 2007. *Object-Oriented Software Engineering*. McGraw-Hill. New York.
- Schamber, L. 2000. Time-line interviews and inductive content analysis: Their effectiveness for exploring cognitive behaviours. *Journal of American Society for Information Science*, 51(8), 734 – 744
- Schwaber, K., Beedle, M. 2001. *Agile Software Development with Scrum*, 1st Edition. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Scott Amber + Associates available at www.ambysoft.com/surveys (viewed 11 August 2016)
SDN Orchestration Layer Implementation Considerations.
- Siakas, K., & Georgiadou, E., 2003. The role of commitment for successful software process improvement and software quality management. *The 11th Software Quality Management Conference, SQM 2003* (pp. 101-113), April 23-25.
- Sommerville, I. 2007. *Software Engineering, Eighth Edition*. Addison-Wesley pp 396 – 398.
- Stapleton, J. 1997. *DSDM: The method in practice*. Addison-Wesley, Inc,
- Tangen, S. 2005. "Demystifying productivity and performance," *International Journal of Productivity and Performance Management* 54(1), 34 – 46.

- Tangen, S., 2002. 'A theoretical foundation for productivity measurement and improvement of automatic assembly systems,' Licentiate Thesis, The Royal Institute of Technology, Stockholm.
- Taylor, C., & Gibbs, G, R. 2010. "What is Qualitative Data Analysis (QDA)?", *Online QDA Web Site*, [onlineqda.hud.ac.uk/Intro_QDA/what_is_qda.php]
- Trendowicz and München, 2009, "Factors Influencing Software Development Productivity-State-of-the-Art and Industrial Experiences," *Advances in Computer*, pp. 185 – 214, Elsevier 2009.
- Trochim, W. M. K. 2006 *Research methods knowledge base*. Retrieved 15 August 2016. <http://www.socialresearchmethods.net>.
- Wagner, S., & Ruhe, M. 2008, "A structured review of productivity factors in software development," *Institut für Informatik-Technische Universität München, Technical Report TUM10832*.
- Walsham, G. 1993, *Interpreting Information Systems in Organizations*, Wiley, Chichester.
- Walston, C.E. & Felix, C.P. 1997. A method of programming measurement and estimation. *IBM Systems Journal*, 16(1), 54 -73
- Weinberg 1971. *The Psychology of Computer Programming*. New York: Van Nostrand Reinhold.
- Whitworth, E., & Biddle, R. 2007. *The Social Nature of Agile Teams*, Agile, Washington, DC, p26 – 36.
- Womack, J.P., Jones, D.J., & Roos, D., 1990. *The Machine that Changed the World*, Rawson Associates, New York, NY.
- Yin, R, K. 2002. *Case study research: Design and methods*. Thousand Oaks, CA: Sage Publications
- Yin, R.K. 2003. *Case study research: design and methods, 3rd edition*. Sage Publications

Yin, R.K. 2008. *Case study research: design and methods*, Applied Social Research Methods Series Vol 5. Sage Publications, 4th edition.

Zimmermann, T., 2017, July. Software productivity decoded: how data science helps to achieve more (keynote). In *Proceedings of the 2017 International Conference on Software and System Process* (pp. 1-2). ACM.