

**A DECISION SUPPORT SYSTEM FRAMEWORK FOR TESTING AND
EVALUATING SOFTWARE IN ORGANISATIONS**

by

TEFO GORDON SEKGWELEO

Thesis submitted in fulfilment of the requirements for the degree

Doctor of Philosophy: Informatics

in the Faculty of Informatics and Design

at the Cape Peninsula University of Technology

Supervisor: Prof T. Iyamu

Cape Town

12 July 2018

CPUT copyright information

This thesis may not be published either in part (in scholarly, scientific or technical journals), or as a whole (as a monograph), unless permission has been obtained from the University

DECLARATION

I, Tefo Gordon Sekgweleo, declare that the contents of this thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed

Date

ABSTRACT

Increasingly, organisations in South African and across the world rely on software for various reasons, such as competitiveness and sustainability. The software are either developed in-house or purchased from the shelf. Irrespective of how the software was acquired, they do encounter challenges, from implementation to support, and use stages. The challenges sometimes hinder and are prohibitive to processes and activities that the software is intended to enable and support.

Majority of the challenges that are encountered with software are attributed to the fact that they were not tested or appropriately tested before implementation. Some of the challenges has been costly to many organisations, particularly in South Africa. As a result, some organisations have been lacking in their efforts toward growth, competitiveness and sustainability. The challenges manifest from the fact that there are no testing tools and methods that can be easily customised for an organisation's purposes. As a result, some organisations adopt more tools and methods for the same testing purposes, which has not solved the problem, as the challenges continue among South Africa organisations.

Based on the challenges as stated above, this study was undertaken. The aim was to develop a decision support system framework, which can be used for software testing by any organisation, owing to its flexibility for customisation. The interpretivist and inductive approaches were employed. The qualitative methods and the case study design approach were applied. Three South African organisations, a private, public and small to medium enterprise (SME) were used as cases in this study. A set of criteria was used to select the organisations. The analysis of the data was guided by two sociotechnical theories, actor network theory (ANT) and diffusion of innovation (DOI). The theories were complementarily applied because of their different focuses.

The actor network theory focuses on actors, which are both human and non-human, heterogeneity of networks, and the relationship between the actors within networks. This includes the interactions that happen at different moments as translated within the heterogeneous networks. Thus, ANT was employed to examine and gain better understanding of the factors that influence software testing in organisations. The DOI focuses on how new (fresh) ideas are diffused in an environment, with particular focus on innovation decision process, which constitute five stages: knowledge, persuasion, decision, implementation and confirmation.

Findings from the data analysis of the three cases were further interpreted. Based on the interpretation, a decision support system framework was developed. The framework is intended to be of interest to software developers, software project managers and other stakeholders, most importantly, to provide guide to software testers in their tasks of testing software. Thus, this research is intended to be of interest and benefit to organisations and academic through its theoretical, practical and methodological contribution as detailed in the chapter seven (conclusion).

In conclusion, even though this research is rigorous, comprehensive and holistic, there are room for future studies. I would like to propose that future research should be in the areas of measurement of software testing. Also, sociotechnical theories like structuration theory and technology acceptance model should be considered in the analysis of such studies.

ACKNOWLEDGEMENTS

I wish to thank the following:

God

I would like to thank God Almighty for giving me life, wisdom, strength, love and respect. The fear of the Lord is the beginning of wisdom. I would like to encourage everyone to welcome strangers in their homes because they may welcome angels without knowing.

Supervisor

I would like to thank my supervisor Professor Tiko Iyamu for the guidance and supervision he provided before and during the research. May the good Lord increase your days, increase your research knowledge, so that you can keep on helping those who aspire to be researchers. We are blessed to have someone like you who is able to carve a rough diamond into a shining stone.

CPUT

I would like to thank the Department of Information Technology (IT) at CPUT for allowing and supporting me in carrying out this research.

Family

I would like to thank my late wife (Moipone Sekgweleo) for her support while she was still living. Mogatsake, you have no replacement; I will always love you and may your soul rest in peace. I am taking care of our boys (Botshelo and Boitsheko Sekgweleo). You are always in our minds and we miss you tremendously. My late father, thank you for encouraging me to study. I listened to you and I am still studying. My late mother, thank you for raising a gentleman like me. You will always be remembered. My mother-in-law, I thank you for giving your daughter in marriage to me. May their souls rest in peace.

DEDICATION

This research is dedicated to:

- Mr Andries Kgarubane Sekgweleo, my late father, for encouraging me to study (May his soul rest in peace)
- Mrs Bafedile Priscilla Sekgweleo, my late mother, for always caring for me (May her soul rest in peace)
- Mrs Sinah Moipone Sekgweleo, my late wife, my other half, my love, the mother of my two sons (May her soul rest in peace)
- Mrs Diatlana Motoma, my late mother in law, my other mother (May her soul rest in peace)
- Botshelo Sekgweleo and Boitsheko Sekgweleo, my two lovely boys who supported me
- My sister, Jeany Sekgweleo, and my brothers, Kagiso Sekgweleo and Utlwanang Sekgweleo
- My late brothers Kgosietsile, Mosimanegape, Diane (Selapi), Kagisonyane, Modise (May their souls rest in peace)
- My colleagues: Annatjie Mogaladi, Kholo Motadi, Mogomotsi Mokoka, Lungelo Boilane, Lerato Matsapola, Mamphofore Ntabane and Mimi Moatshe
- My friends: Solomon Makgale, Monaisa Kgajane and Jerry Molale

You will always be remembered, especially those who have passed on. Those who are still alive, let's enjoy the fruits of my success.

TABLE OF CONTENTS

DECLARATION.....	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vii
CHAPTER ONE	13
INTRODUCTION	13
1.1 Introduction.....	13
1.2 Background of research problem	14
1.3 Problem statement	14
1.4 Literature review.....	15
1.4.1 Software development	15
1.4.2 Software testing	16
1.4.3 Software testing methods.....	17
1.4.4 Software testing tools.....	18
1.4.5 Decision support system	19
1.4.6 Theories underpinning the study.....	20
1.4.6.1 Actor Network Theory	20
1.4.6.2 Diffusion of Innovation.....	21
1.5 Research objectives	23
1.6 Research questions	23
1.7 Research design and methodology	23
1.7.1 Research paradigm	23
1.7.2 Epistemology.....	24
1.7.2.1 Interpretivism.....	24
1.7.3 Research methods	25
1.7.3.1 Qualitative research methods	25
1.7.3.2 Exploratory research methods	26
1.7.4 Research design	26
1.7.4.1 Case study	26
1.7.5 Data collection.....	28
1.7.5.1 Interviews	28
1.7.5.2 Documentation	29
1.8 Data analysis.....	29
1.8.1 Unit of analysis.....	30
1.9 Delineation of the research	31
1.10 Significance of the research.....	31
1.11 Ethical considerations	31
1.12 Structure of the thesis	32
1.13 Summary.....	33
CHAPTER TWO.....	35
LITERATURE REVIEW	35
2.1 Introduction.....	35
2.2 Information technology.....	35
2.3 Software development	36
2.4 Software implementation.....	38
2.5 Software testing	39
2.6 Software testing process.....	41
2.7 Software testing methods.....	43
2.7.1 Black box method.....	43
2.7.2 White box method	44
2.7.3 Grey box method.....	45
2.8 Software testing tools.....	46
2.9 Decision support system	48
2.10 Underpinning theories	50
2.10.1 Actor network theory	50
2.10.2 Moments of translation.....	52
2.10.3 ANT and information systems	53
2.10.4 Diffusion of innovation.....	54

2.10.5	Innovation-decision process.....	56
2.10.6	DOI and information studies.....	56
2.10.7	ANT and DOI.....	57
2.11	Summary.....	58
CHAPTER THREE.....		60
RESEARCH METHODOLOGY.....		60
3.1	Introduction.....	60
3.2	Philosophical assumption.....	60
3.2.1	Epistemology.....	60
3.2.2	Ontology.....	61
3.2.3	Methodology.....	62
3.2.4	Axiology.....	62
3.2.5	Doxology.....	63
3.3	Paradigm taxonomies.....	64
3.3.1	Positivism.....	64
3.3.2	Realism.....	64
3.3.3	Interpretivism.....	64
3.4	Research approach.....	65
3.4.1	Deductive approach.....	66
3.4.2	Inductive approach.....	66
3.5	Research methods.....	66
3.5.1	Quantitative methods.....	66
3.5.2	Qualitative methods.....	67
3.5.3	Mixed methods.....	68
3.6	Research design.....	68
3.6.1	Action research design.....	69
3.6.2	Grounded theory.....	69
3.6.3	Ethnography.....	69
3.6.4	Survey.....	70
3.6.5	Case study.....	70
3.7	Data collection.....	71
3.7.1	Observation.....	72
3.7.2	Experiment.....	72
3.7.3	Interview.....	72
3.7.4	Field work.....	73
3.7.5	Documentation.....	78
3.8	Data analysis.....	79
3.8.1	Unit of analysis.....	80
3.9	Ethical consideration.....	81
3.10	Summary.....	81
CHAPTER FOUR.....		82
CASE STUDY OVERVIEW.....		82
4.1	Introduction.....	82
4.2	Overview: Mootledi Logistics.....	82
4.2.1	Organisational structure.....	83
4.2.2	IT structure.....	84
4.2.3	Roles and responsibilities.....	85
4.3	Overview: Mmuso Technologies.....	86
4.3.1	Organisational structure.....	86
4.3.2	IT structure.....	88
4.3.3	Roles and responsibilities.....	89
4.4	Overview: Bokamoso Solutions.....	89
4.4.1	Organisational structure.....	91
4.4.2	IT structure.....	91
4.4.3	Roles and responsibilities.....	92
4.5	Summary.....	93
CHAPTER FIVE.....		94
DATA ANALYSIS.....		94
5.1	Introduction.....	94
5.2	Overview of data analysis.....	94

5.3	Case study one: mootledi logistics	95
5.4	Actor network theory	95
5.4.1	Actors	95
5.4.2	Networks	96
5.4.3	Moments of translation	98
5.4.3.1	Moments of translation: problematisation	99
5.4.3.2	Moments of translation: interessement	101
5.4.3.3	Moments of translation: enrollment	102
5.4.3.4	Moments of translation: mobilisation	105
5.5	Diffusion of innovation	106
5.5.1	Innovation decision process	106
5.5.1.1	Innovation decision process: knowledge	107
5.5.1.2	Innovation decision process: persuasion	109
5.5.1.3	Innovation decision process: decision	110
5.5.1.4	Innovation decision process: implementation	111
5.5.1.5	Innovation decision process: confirmation	112
5.6	Case study two: mmuso technologies	113
5.7	Actor network theory	113
5.7.1	Actors	113
5.7.2	Networks	115
5.7.3	Moments of translation	116
5.7.3.1	Moments of translation: problematisation	116
5.7.3.2	Moments of translation: interessement	118
5.7.3.3	Moments of translation: enrolment	120
5.7.3.4	Moments of translation: mobilisation	121
5.8	Diffusion of innovation	122
5.8.1	Innovation decision process	122
5.8.1.1	Innovation decision process: knowledge	122
5.8.1.2	Innovation decision process: persuasion	123
5.8.1.3	Innovation decision process: decision	124
5.8.1.4	Innovation decision process: implementation	125
5.8.1.5	Innovation decision process: confirmation	126
5.9	Case study three: bokamoso solutions	127
5.10	Actor network theory	127
5.10.1	Actors	127
5.10.2	Networks	128
5.10.3	Moments of translation	129
5.10.3.1	Moments of translation: problematisation	130
5.10.3.2	Moments of translation: interessement	132
5.10.3.3	Moments of translation: enrolment	133
5.10.3.4	Moments of translation: mobilisation	134
5.11	Diffusion of innovation	135
5.11.1	Innovation decision process	135
5.11.1.1	Innovation decision process: knowledge	136
5.11.1.2	Innovation decision process: persuasion	137
5.11.1.3	Innovation decision process: decision	138
5.11.1.4	Innovation decision process: implementation	139
5.11.1.5	Innovation decision process: confirmation	140
CHAPTER SIX		141
FINDINGS AND INTEPRETATION		141
6.1	Introduction	141
6.2	Findings and discussions: mootledi logistics	141
6.2.1	Lack of testing framework	142
6.2.2	Lack of management buy-in	143
6.2.3	Network of employees	143
6.2.4	Quality of software	144
6.2.5	Lack of standards and procedures	145
6.3	Findings and discussions: mmuso technologies	145
6.3.1	Software evaluation	146
6.3.2	Process oriented	147

6.3.3	Implementation policy	149
6.3.4	Change management.....	149
6.3.5	Power relationship.....	150
6.3.6	Organisational structure	151
6.4	Findings and discussions: bokamoso solutions	152
6.4.1	Heterogeneity of software testers	153
6.4.2	Outsourcing	154
6.4.3	Documentation	155
6.4.4	Queuing system	156
6.4.5	Standardisation	156
6.4.6	Procedural	157
6.5	Decision support system framework for testing and evaluating software	157
6.5.1	Requirements	158
6.5.2	Methodology.....	160
6.5.3	Filtering.....	160
6.5.4	Repository	161
6.5.5	Governance.....	162
6.5.6	Assessment.....	163
6.5.7	Institutionalisation.....	164
6.6	Summary	165
CHAPTER SEVEN.....		166
CONCLUSIONS AND RECOMMENDATIONS		166
7.1	Introduction.....	166
7.2	Summary of the study	167
7.3	Evaluation of the study.....	169
7.4	Contribution of the research.....	175
7.4.1	Theoretical contribution	175
7.4.2	Methodical contribution	176
7.4.3	Practical contribution	176
7.5	Recommendations	177
7.5.1	Documentation	177
7.5.2	Standards and procedures.....	177
7.5.3	Quality of software	178
7.6	Benefit of the study	178
7.7	Further study	179
7.8	Conclusion.....	179
BIBLIOGRAPHY/REFERENCES.....		181
APPENDICES		196

LIST OF FIGURES

Figure 1.1: Moments of translation (Adapted from Rhodes, 2009).....	21
Figure 1.2: DOI stages (Adapted from Nemutanzhela & Iyamu, 2011)	22
Figure 2.1: V-Model (Adapted from Skidmore, 2006)	42
Figure 2. 2: Innovation-decision process (Adapted from Nemutanzhela & Iyamu, 2011)	55
Figure 3.1: The research 'onion' (Adapted from Saunders et al., 2007).....	63
Figure 4.1: Mootledi Logistics organisational structure.....	84
Figure 4.2: IT structure of Mootledi Logistics	84
Figure 4.3: Mmuso Technologies organisational structure	87
Figure 4.4: IT structure of Mmuso Technologies	88
Figure 4.5: Bokamoso Solutions organisational structure	91
Figure 4.6: IT structure of Bokamoso Solutions.....	91
Figure 6.1: Factors influencing software testing and evaluation.....	142
Figure 6.2: Factors influencing software testing and evaluation.....	146
Figure 6.3: Factors influencing software testing and evaluation.....	153
Figure 6.4: Decision support system framework for testing and evaluating software.....	158

LIST OF TABLES

Table 2.1: Software testing levels (Hooda & Chhillar, 2015)	41
Table 2.2: Black box techniques (Hussain & Singh, 2015).....	44
Table 2.3: White box techniques (Khan & Khan, 2012).....	45
Table 2.4: Grey box techniques (Acharya & Pandya, 2012).....	46
Table 2.5: Software Testing Tools (Sharmila & Ramadevi, 2014).....	48
Table 2.6: ANT Tools (Iyamu & Sekgweleo, 2013).....	53
Table 2.7: Innovation-decision process (Sang & Tsai, 2009)	56
Table 3.1: Participants	75
Table 3.2: Units of analysis	80
Table 5.1: Moments of translation (Case 1).....	99
Table 5.2: Moments of translation (Case 2).....	116
Table 5.3: Moments of translation (Case 3).....	130

APPENDICIES

APPENDIX A: Interview Questions	196
APPENDIX B: Ethical Consideration Letter	197

GLOSSARY

Terms/Acronyms/Abbreviations**Definition/Explanation**

Software	Computer program written to be used by organisation
Software developer	A person responsible for developing software
Software tester	A person responsible for testing and evaluating software
Software testing tools	Tools used for software testing
SDLC	System Development Life Cycle
DSS	Decision Support System
ANT	Actor Network Theory
DoI	Diffusion of Innovation
IT	Information Technology

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Organisations increasingly rely on software for their processes and objectives, which makes quality critical. Thus, software testing is an essential aspect of systems development cycle (SDLC) towards fulfilling business requirements and objectives. Thus, software testing is performed by following software quality standards to identify defects and flaws in the software and ensure they are removed (Sowunmi, Misra, Fernandez-Sanz, Crawford & Soto, 2016). Therefore, organisations require skilled personnel, software testers, who understand various types of testing as well as the right tools that enable them to properly conduct software testing and evaluation. Through the use of testing tools, software testers generate, manage and execute the testing of software in a specific environment maintained for a particular type of test (Bhardwaj, 2015). According to Abbas, Sultan and Bhatti (2017) with the automation of software testing tools, software testers can reliably test the software in less time and repeatedly reuse those tools to retest the software.

Organisations invest much money to purchase these tools which simulate what the software testers do manually in an effort to increase efficiency and productivity. The purpose of automated testing is to decrease the number of test cases which could be executed manually with the intention of completely eradicating manual testing (Gautam & Nagpal, 2016). It enables software testers to cover as many test scenarios as possible within a short period of time, saving organisations time and money as compared to performing manual testing. Therefore, it is important for management to be knowledgeable of various software testing tools on the market which they could purchase. They must be in a position to make sound and well-informed decisions pertaining to which software testing tools to purchase.

This chapter, divided into twelve main sections, discusses the overview of this study. The first and second sections discuss the background of the research as well as the problem statement. The third section covers the literature review relevant to this study. The fourth and fifth sections discuss the research objectives and questions. The sixth section covers the research design and methodologies that were followed in the study. The seventh section explains the data analysis and units of analysis. While the eighth, ninth and tenth sections cover the delineation of the research, significance of the research and ethical considerations, respectively. The eleventh

section outlines the structure of the thesis. And finally, the last section concludes the chapter.

1.2 Background of research problem

The research problem emanates from the researcher's own experience working with numerous organisations in South Africa, including financial institutions such as First National Bank, Standard Bank, Telecommunication, Telkom, and the energy and power supply company, ESKOM. These organisations were lacking in testing tools or tools capable of covering end-to-end testing, resulting in a negative impact on the quality of software developed in-house. The poor quality of software is due to lack of, or improper testing that has been a challenge for long time. Within the South African context, Scott, Zadirov, Feinberg and Jayakody (2003) attribute the challenges of software to lack of curriculum coverage in institutions of higher learning. This unfortunately manifests in poor and challenging services provided to consumers. According to Farooq and Quadri (2013), potential failures of software can be avoided by performing exhaustive system testing, allowing possible permutations of inputs which can be either valid and invalid. Some organisations, therefore, sought the use of automated testing tools to address these challenges.

Through automated testing tools, tests can be effortlessly rerun as many times as possible, alleviating human error during software testing (Mishra & Pradhan, 2012). However, the problem remains as there is no framework to guide how the tools can be used, resulting in human error. It is vital for organisations to have a framework in place to guide complementary use of multiple tools when carrying out software testing.

1.3 Problem statement

Software is used to enable and support organisations' processes and activities for competitiveness and sustainability. Thus, software is critical in organisations. As a consequence, it is important to ensure that software is properly implemented, to most effectively address organisational needs. To this end, organisations make use of different tools and methods in testing software to ensure quality and appropriateness for organisational needs.

However, the tools and methods do not allow the organisation to perform multiple testing activities (manual, automation and performance testing), using a single tool. As a result of tool limitations, the organisation fails to achieve its objectives in

conducting end-to-end testing. This has, for many years, caused severe problems for some organisations, affecting their time to respond to business change. The primary problem is that there is no framework to guide complementary use of multiple tools when carrying out software testing.

1.4 Literature review

A *literature review* presents the existing knowledge that has been explored and reviewed by other researchers. Other researchers utilise this existing literature relating to studies to support claims made. This is because literature review assists to identify gaps in a study (Denney & Tewksbury, 2012). Thus, the following sections were discussed: software development, software testing, software testing methods, software testing tools and decision support systems, as well as actor network theory (ANT) and diffusion of innovation (DoI) theory.

1.4.1 Software development

Software development is a process of developing a software system following a particular system development methodology. Scott et al. (2003) explain the full circle of *systems development life cycle* (SDLC) from the basis of its components, and how the objectives are integrated with each other. Along the same line of understanding, Sekgweleo (2015a) describes the SDLC as a practical and systematic process adopted by software developers in developing software enabling the software development team to plan, develop and control the way in which software is developed. These methodologies are comprised of various stages or phases. According to Bukhari, Faisal and Hira (2014), SDLC consists of various stages including the collection of requirements, design, development, testing and implementation. SDLC methodologies can be either traditional or agile.

The phases within the *traditional methodology* follow each other sequentially and the preceding phase has to be completed before starting with the next phase (Bassil, 2012). On the other hand, *agile methodology* is a subcategory of iterative and evolutionary ways that are based on iterative enhancement and adaptable development processes. Kannan, Jhajharia and Verma (2014) allude that the agile development promotes closer relationship with customers in order for them to assess the software and provide continuous feedback. Sekgweleo (2015b:18) describes *agile methodology* as “subset of iterative and evolutionary methods that are based on iterative enhancement and opportunistic development processes”. This methodology is feedback driven with less documentation, with the aim of delivering modules of

software that are functional. Both methodologies have their own strengths and limitations. Agile, for example, makes it possible for software developers to develop software that satisfies users through perpetual delivery of functioning software by getting feedback from the users (Shrivastava & Date, 2010). Van Dijk (2011) argues that due to the little documentation produced in the agile methodology, the artefact developed requires continuous redesign as agile focuses on solving specific problems.

The *waterfall model* is documentation intensive and software development planning has to be completed prior to the actual software development (Alshamrani & Bahattab, 2015). As a result, quality of the software becomes a concern. Both the requirements and technology continuously change; therefore, if planning continues for too long, by the time the software development is completed, the technology could very well be outdated. Also, the requirements might have changed. According to Kannan, Jhajharia and Verma (2014), the waterfall methodology does not tolerate uncertainty and risk due to lack of feedback from business thereby rendering it difficult to improve on the early deficiencies that might have occurred during planning. As a result, poor quality software will be delivered. Therefore, the above mentioned risks needs to be mitigated and managed properly to achieve good quality.

1.4.2 Software testing

Software testing occurs throughout the development or enhancement of existing software. Hooda and Chhillar (2015) describe software testing as a process in which both software requirements and components are tested manually or through the use of software automation tools to determine if the software meets the specified user requirements. Software is evaluated with the intention of producing quality software. Organisations rely on software for competitiveness and sustainability as it enables business to smoothly execute its functions without any disruptions.

It is vital to have clear business requirements prior to software development, testing and evaluation to deliver the quality software required. According to Lee (2014), quality software has to meet various quality factors such as ease of use, user interface aesthetics, functional appropriateness, accuracy and error tolerance. Problems relating to user requirements, when determined late in the software development process, often negatively impact the software cost (Sener & Karsak, 2012). These problems might be realised as early as the planning stage if the software testing is involved early in the software development life cycle. Software

inspection enables the software testing team to systematically detect defects in all stages of software development (Qazi, Shahzadi & Humayun, 2016). This practice assists in exposing defects early so they can be fixed on time. It is much less expensive to correct defects sooner rather than later in the software development life cycle. There are various software testing methods that can be employed to test the software.

1.4.3 Software testing methods

Software testing methods are basically the approach that can be adopted to test and evaluate software within the organisation, pointing out which direction to follow when conducting software testing. According to Mishra, Ostrovska and Hacaloglu (2017), the two commonly used types of testing methods are black box (functional) and white box (structural) testing. In *black box testing*, the software is tested upon expected output; the tester does not need to know the internal workings of the software (Dhiman & Sharma, 2016). When using this method, software testers are only mindful of what the software is supposed to do. This testing is purely based on the requirement specification knowledge (Nidhra & Dondeti, 2012). As a result, when the actual result from the software does not correspond with the expected result from the requirement specification, a defect is logged.

In *white box testing* the software tester tests both the functionality as well as the internal workings of the software (Jamil *et al.*, 2016). With this kind of testing, the software tester has the exceptional knowledge of how the software functions. According to Nidhra and Dondeti (2012), the software tester is granted access to the code when performing white box testing to be able to test the code of the software. As the white box testers perform testing beyond the user interface, they are able to detect defects from the code perspective. For example, enterprise resource planning (ERP) systems are complex to implement in large organisations as compared to their medium or small counterparts (Muscatello, Small & Chen, 2003). Goyette, Cassivi, Courchesne and Elia (2015) highlighted that recent statistics indicate that more than 50% of projects experience cost overruns and more than 60% have schedule overruns and these numbers have hardly changed since 15 years ago, when 70% of ERP implementations were considered to be failures. Therefore, in order to deliver quality software, rigorous testing of such software is required.

More recently, the third method that has been introduced is known as *grey box testing*. In grey box testing, the software tester has limited knowledge about the

internal workings of the software (Jan, Shah, Johar, Shah & Khan, 2016). This combines the strengths of both white box and black box testing (Khan & Khan, 2012). With this kind of testing the software tester is not required to have full access of the software's source code. Each of the above testing methods can be used in conjunction with various software testing tools.

1.4.4 Software testing tools

Software testing can be conducted manually and also through automation testing tools. Manually, software testers create test cases and execute them through a manual testing tool. However, this process is tedious, and it is easy for software testers to commit mistakes. Bamotra and Randhawa (2017) argue that manual testing is time consuming, resource intensive and allows some defects to remain uncovered. Automation tools, then, enable automation testers to record and replay those test cases or user actions which could be performed manually by a software tester (Singh & Tarika, 2014). With automated testing, the software testing is capable of reducing time, cost and productivity because automation test scripts can be run repetitively for the software under test. Whenever the software is enhanced, or some functionality changed, the automation test scripts are also modified in order to be re-used in testing the software.

Software automation tools can be either proprietary or free open source. The proprietary software testing tools are commercialised and require licensing per use. Free open source tools, on the other hand, are free and downloadable from the Internet. According to Singh and Tarika (2014), the free open source automation software testing tools do not require any licences to be purchased for use and the software code is available to the user for further enhancements to be done. Some of the open source automation tools include Apache Selenium, Geb, Windmill, GitHub Protractor, SpecFlow, Tyto Software Sahi and BSDW (Saravanan & Prasad, 2016). Commercial automation testing tools include HP Unified Functional Testing, IBM Rational Functional Tester, Oracle Application Testing Suite, Borland Micro Focus SilkTest, SmartBear Test Complete and Testing Anywhere (Waje, Gaikwad & Chaudhari, 2014). Many organisations purchase proprietary software testing tools because they are ready to be used (Monier & El-mahdy, 2015). While with open-source testing tools are freely downloadable from the internet however, further development is often needed, which some organisations are usually not prepared to undertake (Singh & Tarika, 2014).

Additionally, there are performance software testing tools which are both open source and proprietary. According to Abbas *et al.* (2017) currently there are various open source and commercial load testing tools available in the market, including LoadRunner, Apache JMeter, LoadRunner, Siege and Microsoft Visual Studio (TFS). Clearly, organisations have variety of software testing tools from which to choose.

1.4.5 Decision support system

Managers within organisations are faced with various decisions to make, some difficult and others not so difficult. The decisions they make, however, become a final choice, without any frame of reference, which is problematic for many organisations in South Africa. Thus, support systems are needed to guide the decision of the managers and software testers. However, most current frameworks and methodologies were created by European countries, which are not necessarily suitable for the South African environment (De Wet & Visser, 2013). Currently, systems such as *decision support* are available to assist managers in making decisions which enable the organisations to continue functioning. Decision Support Systems (DSS) are computer-based information systems designed to assist managers in making informed decisions when faced with problems (Tripathi, 2011). According to Jain and Raju (2016:42), "DSSs serve the management, operations, and planning levels of an organisation and help to make decisions, which may be rapidly changing and not easily specified in advance".

At times it is difficult for humans to make decisions, especially if the root cause of a problem is not well-understood. Thus, Filip, Zamfirescu and Ciurea (2017), emphasising that for decision-makers to overcome limits and constraints encountered, explain that they may need to rely on DSS to assist in making difficult decisions to solve complex problems. Decision-making is one of the essential activities of management and is a huge part of any process of implementation. According to Liu, Duffy, Whitfield and Boyle (2010), various DSSs were developed to support decision makers at all levels in the organisation, including systems that could support problem structuring, operations, financial management and strategic decision making, even extending to support for optimisation and simulation. Even in software testing and evaluation, DSS can be used to assist managers in making right decisions.

1.4.6 Theories underpinning the study

This study aimed at developing a decision support system framework for testing and evaluating software in organisations. The study was underpinned by two theories – actor network theory (ANT) and Diffusion of Innovation (DOI) – meaning that the theories guided the study from two different perspectives.

1.4.6.1 Actor Network Theory

The actor network theory (ANT), originating from the field of sociology, focuses on performance relations between human and non-human actors and how they connect in formulation of socio-technical networks with aligned interests (Effah, 2012). Both human and non-human actors contribute equally to networks. Luoma-aho and Paloviita (2010) assert that humans are not the only entities that act with agency: all actors (including objects) play equal roles within the network. Each actor has something to contribute for the network to be functional. Teles and Joia (2011) further alluded that actor-network theory is a combination of agency and structure, as none of them, actor or network, exists independently of the other.

The two complement each other in a way that if one is absent, the actor network becomes dysfunctional. It is vital to understand the entire concept of actor network. Williams-Jones and Graham (2003) emphasised that in order for us to distinguish the origins of power and structure in the actor network, we need to consider all the components that collaborate, co-operate, compete and lead to creation, persistence or perishing of that network. Mahring, Holmstrom, Keil and Montealegre (2004) explained that the actor-network creation, also referred to as *translation*, is comprised of four major stages: problematization, interessement, enrollment and mobilization. These four moments of translation were used as a lens to zoom into the collected data. The diagram below depicts the four moments of translation of ANT.

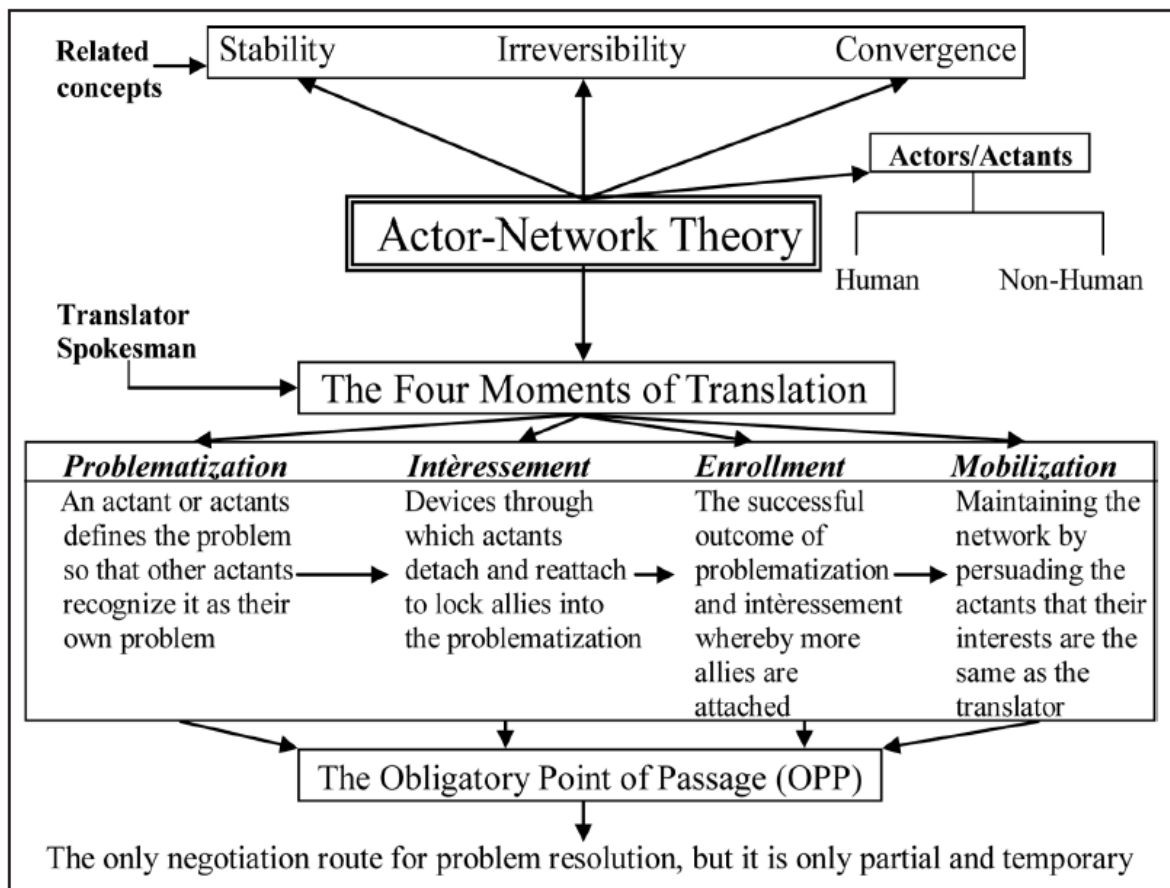


Figure 1.1: Moments of translation (Adapted from Rhodes, 2009)

The actor network theory suggests that knowledge is created, though the creation is as the result of a heterogeneous/diverse network of people, devices and texts which make a form of balance (Steen, 2010). ANT was used as a lens to zoom into actors involved in the evaluation and testing of software within the three organisations.

1.4.6.2 Diffusion of Innovation

The process from the diffusion of innovation (DOI) theory was also employed in the data analysis for this study. The DOI theory, introduced by Rogers in 1962, is defined as a process by which innovation of technologies is communicated to the members of social system through certain communication channels to masses over time (Chang, 2010). This theory is concerned with introducing new ideas or technologies to the target market. With new technology being introduced to an organisation, it is evident that resistance from software users usually occurs. Software users may not like the new software (because it has new features, is difficult to use, takes time to read user manuals) and prefer the older one (because there is nothing new to learn, they know it by heart) or they just don't care to change. Therefore, DOI focuses on diffusing new ideas to the environment, with particular focus on innovation decision

process, which constitute five stages: knowledge, persuasion, decision, implementation and confirmation.

Rogers also identified five significant characteristics of the innovation that influences its adoption: namely relative advantage, compatibility, complexity, trialability, and observability (Olsson, Skovdahl & Engström, 2016).

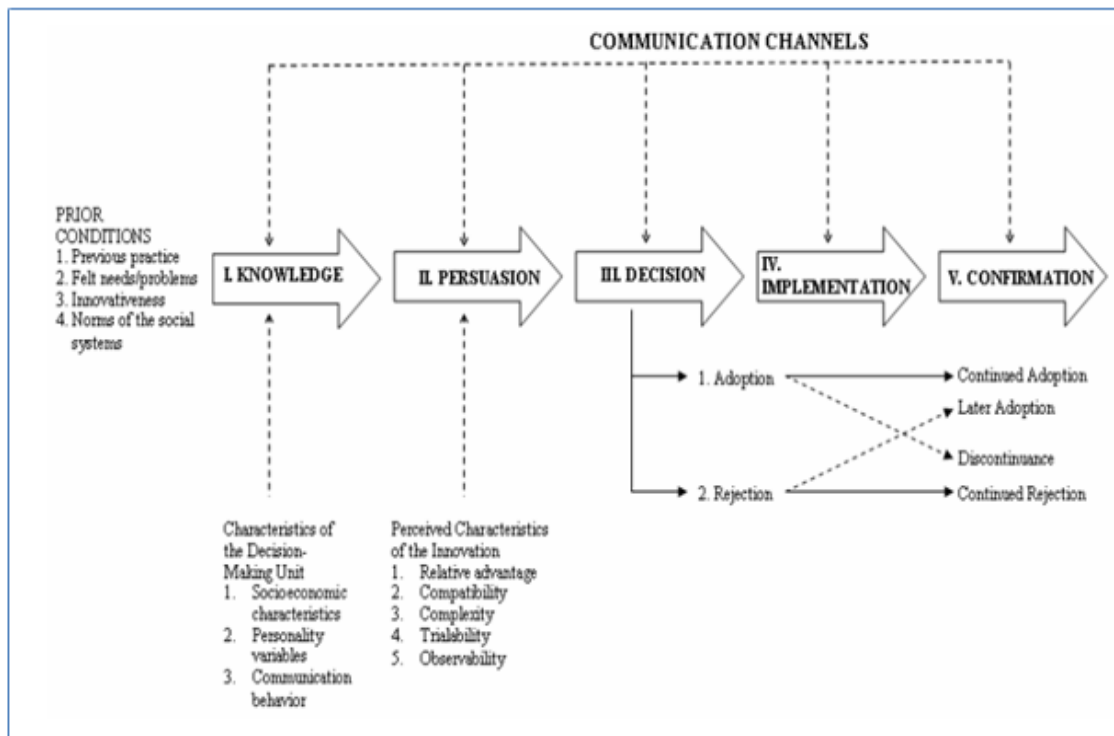


Figure 1.2: DOI stages (Adapted from Nemutanzhela & Iyamu, 2011)

Relative advantage describes how potential adopters expect the innovation to improve their lives (Montfort, Brown & Pegg, 2009). This new idea or concept is expected to bring change and simplify how things are done within the organisation. *Compatibility* refers to a point which the innovation is seen as reliable with current values, previous experiences and requirements of potential adopters (Zolkepli & Kamarulzaman, 2015). *Complexity* refers to a point at which the innovation is considered difficult to understand, implement and use (Loukis, Charalabidis & Androutsopoulou, 2017). *Trialability* is the point at which the innovation may be tested on a limited basis before making an adoption (or rejection) decision (Ekdale *et al.*, 2015). And finally, *observability* is the point at which the results of the innovation are visible to others (Olsson *et al.*, 2016). Both ANT and DoI theories were adopted in this study.

1.5 Research objectives

The aim of this present research is to develop a decision support system framework for testing and evaluating software. The framework is intended to help guide organisations in complementary use of multiple tools when carrying out software testing. The objectives are as follows:

- i. examine and understand the tools (for manual, automation and performance testing) used for software testing;
- ii. explore and understand methods (approaches adopted for testing such as white or black box testing) involved in testing the software;
- iii. examine the factors (factors triggering testing to be conducted) that could influence the testing and evaluating of software in organisations; and then
- iv. based on the findings from the objectives as stated above, a decision support system framework will be created. The aim of this decision support system framework will be for addressing the challenges which occur during software testing and evaluation in organisations.

1.6 Research questions

To achieve the above research objectives, four main questions have been formulated:

- i. What are the tools that are used in testing software?
- ii. What are the methods involved in the testing of the software?
- iii. What factors influence the testing and evaluating of software in organisations?
- iv. How can a decision support system framework be developed for addressing the challenges which occur during software testing and evaluation in organisations?

1.7 Research design and methodology

The purpose of this study was to develop a decision support system framework to test and evaluate software within organisations. In order to achieve this goal, various methods and approaches were employed, including a qualitative research approach, case studies and data collection approaches.

1.7.1 Research paradigm

A *paradigm* is how the world views things. Petersen and Gencel (2013:81) describe paradigm as "a basic set of beliefs that guide action". The word *paradigm* is used interchangeably with the word *philosophy* in the field of research. The aim of a paradigm is not meant to lead people into making conclusions, but to guide them get

where they want to. The paradigm, or worldview, influences the types of research methods that can be adopted by a researcher as part of the research methodology in delivering trustworthy evidence about the examined phenomenon of interest (Brown, 2009).

A paradigm points the researcher in the right direction, keeping the researcher focused on the subject researched. The knowledge of research paradigms helps the researcher to assess various methodologies and methods to avoid incompatible use and unrequired work, confronting the limitations of certain methods at an early stage of the research (Wijesinghe, 2009). There are four major aspects of research paradigms, namely ontology, epistemology, axiology and pragmatic (Ihuah & Eaton, 2013).

1.7.2 Epistemology

Epistemology is something that can be known about what exists. Epistemology focuses on the nature and forms of knowledge (Cohen, Manion & Morrison, 2007:7). According to Scotland (2012), epistemological assumptions emphasise how knowledge is created, acquired and communicated. Saunders, Lewis and Thornhill (2009) further stated that epistemological assumptions are concerned with what is acceptable as knowledge and also that which creates the acceptable knowledge within a field of study. Similarly, according to Tennis (2008), epistemology is how we know, helping to determine whether the knowledge acquired or created is valid. It enables the researcher to analyse the nature of knowledge. Epistemology could follow various paradigms, including post-positivist, constructionist and interpretivist (Levers, 2013).

1.7.2.1 Interpretivism

The *interpretive paradigm* enables researchers to explore their world by interpreting what other individuals know or understand. Cresswell (2007) asserted that with interpretive paradigm, the researcher aims at making sense or interpreting the meanings other individuals have about the world. Thus, the interpretive paradigm emphasises recognising and reciting the meaning of human actions and experiences (Levers, 2013). Interpretivists must understand the context of any type of research conducted and the criticality of the interpretation of collected data (Thanh & Thanh, 2015). The researcher's interpretation thus becomes subjective as it is influenced by feelings and emotions. That is how reliable knowledge gets created. Thus,

interpretivist research is “guided by the researcher’s set of beliefs and feelings about the world and how it should be understood and studied” (Terreberry, 2017:55).

Therefore, the role of the interpretivist researcher is to understand, explain and interpret social reality through a different eye (Mack, 2010). The intention of this present research is to create a decision support system framework for the testing and evaluation of software in an organisation. Therefore, the interpretivism paradigm was employed for this study because of its characteristics. The researcher subjectively interpreted the qualitative data that was collected following the theories underpinning this study (actor network theory and diffusion of innovation). Actor network theory was applied first, followed by diffusion of innovation, primarily because it was critical to first establish the formulation and existence of networks so as to know how the technology can be diffused in the environment. Also, it was necessary to first understand the tools, methods, and relationships between the actors involved in software testing, prior to assessment of innovative and diffusion.

1.7.3 Research methods

Research methodology, a vehicle enabling a researcher to conduct research on a topic of choice, offers a set of methods that can be applied to a particular case. Abu-Dalbouh (2013) describes *research methodology* as procedures, ways, methods and techniques used to collect the required information relating to the research objective. There are a variety of research methods, namely qualitative, quantitative and mixed method, which are selected based on the purpose of the research. For the purpose of this research, qualitative research methodology was employed.

1.7.3.1 Qualitative research methods

Qualitative research aims at explaining a social phenomenon by examining people’s beliefs, culture and experiences. A qualitative approach enables the researchers to capture the thoughts and feelings of the research participants, further enabling the understanding of the meaning that people ascribe to their experiences (Sutton & Austin, 2015). It provides rich descriptive accounts of the phenomenon under investigation (Gelo, Braakmann & Benetka, 2008).

This approach investigates the why and how of the phenomena. Rajasekar, Philominathan and Chinnathambi (2013) characterise this approach as being non-numeric, descriptive, applying reasoning and using words to describe the findings. The justification for using qualitative research was that it has the capability of digging

deeply into peoples' experiences, behaviours, thoughts and beliefs, assisting the researcher in understanding the factors influencing the testing and evaluation of business software. This understanding was established through interviewing participants, encouraging them to share how the software was tested and evaluated within the organisation. The qualitative approach was used in conjunction with the case study.

1.7.3.2 Exploratory research methods

Exploratory research is the method appropriate for investigating new or a problem that has not been studied clearly. The intention of exploratory research is to formulate a problem to precisely investigate or develop working hypotheses from an operational point of view (Kothari, 2004). It allows people to think, to use their imagination, experience, insight and skill to propose innovative ways for understanding and interpreting reality (Reiter, 2013). This is where a researcher has an idea or has observed something and seeks to understand more about it, laying the groundwork for future research. For this particular study, the researcher was seeking to provide knowledge about testing and evaluation of software within various organisations. To gain this broad understanding, and as organisations operate differently, the selected organisations included a private, a public and a small-medium enterprise (SME).

1.7.4 Research design

Research entails a great deal of information searching and reading surrounding the objective of the study. Ragab and Arisha (2018:1) define *research* as “systematic investigation into and study of materials and sources in order to establish facts and reach new conclusions”. These facts expand on the existing knowledge. Therefore a research design is required to guide a researcher when conducting the research. Wedawatta, Ingririge and Amaratunga (2011) argue that a research design offers direction for the research as well as the process by which the research is conducted. The case study approach was employed in this study.

1.7.4.1 Case study

The case study aims at investigating an ‘instance’ which may be an organisation, department, project, information system, a person or even a kind of illness. Yazan (2015:138) defines *case study* as “a contemporary phenomenon within its real-life context, especially when the boundaries between a phenomenon and context are not clear and the researcher has little control over the phenomenon and context”. Yin

(2013) explains that the case study approach enables the researcher to clarify questions such as the "how" and "why" of an event or phenomena. The approach can be used to investigate the phenomenon in its real-life context through the use of one or multiple entities (cases) (Benbasat, Goldstein & Mead, 1987). Through the detailed investigation, such as the use of the case study approach, deeper understanding of the phenomenon can be gained (Nabukenya, 2012). The case study offers a systematic way of viewing events, gathering data, evaluating information and reporting results as it provides detailed contextual views on phenomenon of interest.

Three organisations were selected as case studies for this research to gain a deeper understanding of how software was tested and evaluated within these organisations. The cases consisted of a private (Mootledi Logistics), public (Mmuso Technologies) and small and medium enterprise (SME) (Bokamoso Solutions). The business objectives and focuses of these organisations were different at the time of this research. Mootledi Logistics was in automobile and logistics sector; whilst the other two organisations, Mmuso Technologies and Bokamoso Solutions focused on information technology (IT) as their core business. Organisations are classified in terms of sector, size, total number of employees, total turnover and total gross asset value. A private company is a privately-owned company that is prohibited from offering its shares to the public – the transferability of its shares are restricted – but it may have more than 50 shareholders (Department of Trade and Industry, 2010). A public company is one in which the ownership is distributed among the overall public with many shares of stock which are freely traded on a stock market but restricts their right of pre-emption (Department of Trade and Industry, 2010).

In South Africa, an SME is any enterprise with fewer than 100 to 200 employees, an annual turnover of less than R4 million to R50 million and gross assets, excluding fixed assets, less than R2 million to R18 million depending upon the industry and has direct managerial involvement by owners (Abor & Quartey, 2010). The public organisation has its own way of conducting business, different from the private organisation and the small medium enterprise (SME). All three organisations' operational activities are conducted differently. Hence, the researcher was able to identify how different or similar things were conducted in the mentioned organisations when testing and evaluating software.

1.7.5 Data collection

After the research design, the next step to consider was data collection, achieved by consulting as many relevant participants as possible. Data collection enabled the researcher to systematically collect data about a chosen topic. Data collection methods influence the reliability and validity of the results (Yang, Wang & Su, 2006). According to Oates (2006), data can be collected through various data collection methods such as interviews, questionnaires, observations and documentation. This study employed interviews and documentation as the different approaches in the data collecting process.

1.7.5.1 Interviews

Interviews are widely used in research as a data collection tool. Unlike questionnaires, interviews are more powerful in obtaining rich data that allows researchers to examine people's views in greater depth (Alshenqeeti, 2014). An interview, a conversation that occurs between the interviewer and interviewee, enables the interviewer to explore the meaning of what an interviewee says. Likewise, Harris and Brown (2010) assert that the intention of an interview is to gather in-depth insights on participant attitudes, thoughts and actions. Interviews, then, enabled the researcher to gather more information from the interviewee regarding the investigated topic of software testing and evaluation. There are three different interview approaches: structured, semi-structured and unstructured.

A *semi-structured interview*, is a pre-planned interview where the researcher writes the interview questions prior to conducting the interview, enables the interviewer to allow the interviewee to elaborate on particular issues which are ambiguous (Alsaawi, 2014). This type of interview is a controlled way of obtaining information from interviews. Therefore, semi-structured interviews were employed for this study as the interviewer could prompt the interviewee for further clarity on answers that were unclear.

However, with *structured interviews*, the interviewer is not required to follow the questions as prepared in sequential order but can meander through the questions depending on the flow of the conversation. This means the interviewee is free to respond to questions and even ask questions. This approach allowed the interviewer to take notes during the interview and record the conversation. The recording of the interview enables the researcher to focus on the interview content and verbal

prompts, simplifying interview transcriptions (Jamshed, 2014). A recording captured the entire conversation.

For this study, a voice recorder was used to capture the interview conversation which was later transcribed word by word. The interviews were conducted in English as this is the common language of the information technology field. Permission to record the conversation was requested prior to each interview.

1.7.5.2 Documentation

Documentation is the other data collection tool chosen for this research. Document is a broad term defined to cover variety of written records, physical traces and visual images (Merriam & Tisdell, 2015). Documents can be classified as either 'found' or 'researcher-generated'. According to Kolhe, Khetri and Deshmukh (2013), found documents exist prior to the research and can be found in most organisations, whilst researcher-generated documents are compiled solely for the purpose of the research. These found documents include user manuals, user requirements, functional specifications, minutes and status reports, for example. On the other hand researcher-generated documents include interview notes, interview recordings, field notes, answers to questionnaires, journals and articles, for example.

Access to documents such as software testing tools and methods and strategies related to the study were requested from the organisations and used as supporting documentation for the interviews conducted when discussing areas such as software development, software testing and decision support.

1.8 Data analysis

Data analysis occurs after the completion of data collection. Binckman and Rog (2009:234) state that data analysis is "what does a researcher do with the collected evidence to make sense of it". Two theories – actor network theory and diffusion of innovation – were used as lenses at both macro and micro levels in the analysis of the data. ANT was used at the macro level, mainly because it guided the DoI, which was used at micro level, to diffuse the innovation. The two theories were employed as follows:

First, ANT was applied as follows:

- i. to understand software testing through an examination of the problematisation of tools (e.g. for manual, automated and performance testing) that are used for software testing;
- ii. to examine the relationship between actors to gain understanding of software testing methods, through their selection and adoption; and
- iii. to examine the factors that influence testing and evaluation of software through moments of translation.

This was followed by the use of DOI, as follows:

- iv. to understand how decisions are made in testing software in organisations; and
- v. to explore how software innovation is diffused across the organisations.

The rationale for adopting DOI was that decisions made in software testing were vital. Such decisions could either impact the quality of software negatively or positively. Negative in a sense that when software testing activities (such as which testing methods and techniques to adopt) are not planned and executed properly, poor quality software may be implemented. As a result, business would not be impressed with the software delivered by the software development team. Therefore, the software testing team must be innovative at all times in terms of deciding how the software needs to be tested, what are the anticipated risks, how can they be mitigated, which functionality or test cases needs to be prioritised, which software testing methods and techniques to adopt, whether automation and performance testing is necessary and at what stage can they be carried out. Software testing is allocated time as other activities in the systems development cycle. Therefore, making the right decisions from the onset when the testing team is involved would assist the team in meeting the scheduled timelines without compromising the quality of software. Through innovative ideas quality software would be achieved in testing software.

1.8.1 Unit of analysis

A *unit* is a single thing that can be regarded as complete but individually forms part of a bigger thing, while *analysis* means breaking a complex topic into smaller portions in order to gain better understanding. Bhattacharjee (2012) describes *unit of analysis* as a person, group or object that needs to be explored to be understood better. Elo and

Kynga (2007) further describes unit of analysis as a letter, sentence, and a portion of pages or words, number of participants in discussions or time used for discussion. In this study, three cases were explored, consisting of technical (IT) and non-technical (business) participants.

1.9 Delineation of the research

The study focused on software testing. The physical boundary (where software testing is conducted) was not part of the research. Therefore, for the purpose of this study, software testing referred to manual, automation and performance testing. This study purely addressed the software testing component, not the implementation of the software.

1.10 Significance of the research

The benefits of the study were threefold: theoretical, methodological and practical:

- i. *Theoretical perspective:* studies have been conducted with regard to software testing and evaluation. However, little is known about the factors influencing software testing and evaluation in an organisation. As a result, this study seeks to contribute to the academic body of knowledge, increasing the existing literature.
- ii. *Methodological perspective:* employing two sociotechnical theories, as lenses through which to analyse the data, brings a renewed perspective to how software testing and evaluation has been studied.
- iii. *Practical perspective:* the study will assist decision makers as well as managers in knowing more of what software testing and evaluation is about. Decision makers and managers will understand the challenges which occur during software testing and evaluation in organisations.

1.11 Ethical considerations

The process of this research adhered to the university, Cape Peninsula University of Technology (CPUT) research ethics code of conduct. It is within the code of conduct that anonymity was emphasised in the consent form and at the briefings. Thus, the researcher:

- i. Explained to the participants their right to withdraw from the process at time they deem fit to do so.
- ii. Provided a consent form, requesting respondents to complete it as a guarantee that their identities will not be revealed.

1.12 Structure of the thesis

The research study consists of seven chapters, summarised as follows:

CHAPTER 1: Introduction

This chapter introduces the research topic as documented in the study. It also provides the introduction to the study, including the research problem, research objectives and research questions. This chapter also covers the literature review relating to the study, research methodologies applied and the underpinning theories that were applied in the data analysis and the conclusions reached. Thus, the chapter provides an overview of the entire study, explaining how the thesis is organised.

CHAPTER 2: Literature review

This chapter presents the acknowledgement and recognition of existing studies to support the objectives of the research. Furthermore, it covers the discussion on literature related to the testing and evaluation of software within organisations. The literature review covers six main parts of the study: software development, software testing, software testing methods, software testing tools, decision support system and the theoretical underpinnings of the study. Two theories underpinning the study, actor network theory (ANT) and diffusion of innovation (DOI), were also discussed. The moments of translation, also known as the lens of ANT, was applied. Thereafter, the innovation decision process was applied to diffuse the innovation.

CHAPTER 3: Research methodology

This chapter focused on the research methods and approaches adapted for the study which include research strategy, design and data gathering techniques. The research methodologies, including approaches, methods and techniques that were applied in the study were discussed in this chapter. The researcher applied a qualitative research method, while also opting for a case study and various data collection techniques such as interviews and documentation. The case study research approach was employed in the study, with interviews generated the data to collect from all three different organisations (a private, an SME and a public organisation). The intention was to understand how these three organisations conduct software testing and evaluation in order to develop a decision support system framework for testing and evaluating software.

CHAPTER 4: Case study overview

The overview of three cases used in the study is presented in this chapter, including the goals, strategy and vision of the individual organisations, the organisational structure, and the roles and responsibilities of the departments within the organisations. The three case study interviews were carried out using the same strategy, but the organisations were treated differently since they operate in different businesses (private, SME and public). The businesses chosen as case studies are not in competition with each other as they have different cultural settings, one being in the automobile and logistics sector and others in the information technology (IT) sector.

CHAPTER 5: Data analysis

The analyses and subsequent findings for all three case studies are presented in this chapter. The analyses were carried out using moments of translation from the perspective of the actor network theory as well as the innovation diffusion process from the perspective of the diffusion of innovation theory as discussed in Chapter 2. The stages of the moments of translation were used to analyse the data. Also, the innovation decision process stages were used to analyse data. Actor network theory was used to establish actors (human and non-human) as well as networks involved in the testing and evaluation of software.

CHAPTER 6: Findings and interpretation

The findings and interpretations of this study are presented in this chapter. The findings for each case were explained separately. Then, based on the findings and interpretation, the decision support system framework for testing and evaluating software was developed. The framework is aimed at addressing the challenges which occur during software testing and evaluation in organisations.

CHAPTER 7: Conclusion and recommendations

This final chapter summarises all the previous chapters and provides the evaluation of the study. The theoretical contributions of the study are presented and recommendations and suggestions for further research are made in this chapter.

1.13 Summary

The aim of the study was to create a decision support system framework for testing and evaluating software within organisations. In this chapter, the qualitative research method, together with the case study, was introduced to examine how software

testing and evaluation was performed within three types of organisations: a private, a public and a small medium enterprise, respectively. This was necessary as organisations rely on software for sustainability and competitiveness. Therefore, it is imperative for all software developed to be rigorously tested and evaluated to ensure that business carried on without any interruptions. As such, customers will be delighted to be associated with the organisation that satisfied its customers. This software enables an organisation to function effectively and perform its duties efficiently. Therefore, all software produced needs to be of high quality to retain and even advance the organisation's competitiveness. Customers expect to receive services rendered to them at all times and at their convenience, without faults or disruption. In the next chapter, the review of literature relating to this study that was conducted is presented.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter, discusses the literature relating to this study, and it is divided into nine main parts. The first part covers the information technology (IT). The second part discusses software development and implementation. The third to the sixth part covers software testing, software testing processes, software testing methods as well as software testing tools. The seventh and eighth part, describe the decision support system (DSS) and the theories underpinning this study: the actor network theory (ANT) and the diffusion of innovation (DOI) theory.

2.2 Information technology

Information technology (IT) is the use of computers to store, disseminate or retrieve information. Ghobakhloo, Sabouri, Hong and Zulkifli (2011:54) describe IT as “capabilities offered to organisations by computers, software applications, and telecommunications to deliver data, information, and knowledge to individuals and processes”. IT allows organisations to function more efficiently and thereby maximise productivity and profits. It also provides faster communication, electronic storage and the protection of records within the organisation. Hence, many organisations depend on IT as an enabler to function. According to Sembiring and Adi (2015), IT assists organisations to adapt and improve the quality of their service. However, once implemented, the employees might still decide not to use it. Therefore, organisations need to instigate better ways of enforcing the use IT.

As technology improves, tasks that were previously performed by human employees are now carried out by IT systems. As a result, some employees ultimately lose their jobs due to technology. In other instances, however, employees tend to prefer the manual ways of performing their daily operations even though the organisation has purchased IT systems. Sweis *et al.* (2014) argue that a wide negative perception and strong resistance of employees in using IT may translate to a lack of support as well as a shortage of skills required for using IT systems. The lack of IT knowledge within the organisation can be viewed as a barrier to IT adoption (Ghobakhloo, Hong, Sabouri & Zulkifli, 2012). Therefore, organisations that are relying on IT have to consistently train employees to keep them abreast with the latest technology and to remain efficient and effective. The consequence of not keeping employees up to date is that employees are rendered as unnecessary in the face of changing technologies.

Some organisations purchase or develop software in-house and implement it so long as they have relevant skills in place.

2.3 Software development

Software development is a process of developing a software system following a particular systems development methodology. These methodologies, either traditional or agile, are comprised of a sequence of stages that need to be followed by software developers to produce and deliver software requested by business (Alshamrani & Bahattab, 2015). These methodologies offer discipline to software development processes for efficiently developing the requested software (Bukhari & Khan, 2014). Amlani (2012) further explained that software development life cycle (SDLC) enables the software development team to plan, develop and control the way in which software is developed.

The deliverables on agile methodology occur iteratively based on the functionality prioritised by business. Kannan *et al.* (2014) asserted that agile methodology is a mixture of iterative and incremental process models which promote flexibility and the timely delivery of software. This methodology does not allow software to be developed for reusable purposes, but only focuses on solving specific and not general problems (Sekgweleo, 2015b). The traditional methodology, such as the waterfall method, is documentation intensive and software development planning has to be completed prior to the actual software development (Alshamrani & Bahattab, 2015). As a result, quality of the software becomes a concern.

Both the requirements and technology are continuously changing; therefore, if planning takes too long, by the time the development of software is complete the technology might already be outdated, or the requirements might have changed. According to Kannan *et al.* (2014) the waterfall methodology does not tolerate uncertainty and risk due to lack of feedback from the business. Consequently, it makes it difficult to improve on the early deficiencies that might have occurred during planning.

The phases within the traditional methodology follow each other sequentially and the preceding phase has to be completed before starting with the next phase (Bassil, 2012). The deliverables on agile methodology occur iteratively based on the functionality prioritised by business. Sekgweleo (2015b:18) describes agile methodology as “subset of iterative and evolutionary methods that are based on

iterative enhancement and opportunistic development processes". This methodology is feedback driven with less documentation, with the aim of delivering modules of software that are functional. Both methodologies have strengths and limitations. Agile makes it possible for software developers to develop software that satisfies users through perpetual delivery of functioning software by getting feedback from the users (Shrivastava & Date, 2010). Van Dijk (2011) argue that due to the little documentation produced in agile, the artefact developed requires continuous redesign as agile focuses on solving specific problems.

The waterfall model is documentation intensive and software development planning has to be completed prior to the actual software development (Alshamrani & Bahattab, 2015). As a result, quality of the software becomes a concern. Both the requirements and technology continuously change, so if planning takes too long by the time the development of software is complete the technology might be outdated. Also, the requirements might have changed. According to Kannan *et al.* (2014) the waterfall methodology does not tolerate uncertainty and risk due to lack of feedback from business. Therefore, it makes it difficult to improve on the early deficiencies that might have occurred during planning. As a result, poor quality software will be delivered.

The methodology (agile or traditional) is chosen according to the needs of the project (Vinekar, Slinkman & Nerur, 2006). Therefore, it is imperative to understand that software testing is informed by the type of methodology that is adopted by the software development team. According to Sekgweleo (2015a), it is vital for the stakeholders involved in the software development team to decide wisely on the methodology to be adopted for a particular project because software development methodology is not a silver bullet for all projects. Software development methodology is just a framework to be followed when developing software. Software testing is influenced by the software development methodology adopted by the software development team. Saleh (2011) argued that the intention of software testing is to deliver good quality software to the customer or project outcome as anticipated by all stakeholders.

There are various activities carried out in the development phase which includes database design and creation, user interface design, application, library and system sources and binary code, and the developing and testing of software against the business requirements specification (Iyamu, Sekgweleo & Mkhomazi, 2013). The

creation of user interface and source code requires tools to mention few such as C++, Java, Oracle and Delphi to aid in the aspects of software (Avison & Fitzgerald, 2006). These tools enable the software developers to convert the business requirements into programs that can be used for daily operations within the organisation. Once the software development is complete, software testing commences. Once the software testing is complete and the business has accepted what they requested, then software implementation occurs.

Organisations develop software for sustainability and competitiveness. According to Tarrant (2016:n) from Moneyweb, "South Africa's 'Big Four' retail and commercial banks spent in excess of R30 billion on information technology over the 12 months to 30 June 2016, including the cost of staff involved in this function". Clearly, organisations invest huge amounts of money in IT to improve how they do business and to out-compete their rivals. So software must be tested thoroughly prior to its implementation.

2.4 Software implementation

Software implementation is a process of deploying the tested software into the production environment. Nahas and Maaita (2012) describe software implementation as a way of translating the software specification into the executable software. This simply means that the software has been developed, tested and is ready for use and can be deployed into production environment. However, the implementation phase in some software development methodologies includes development and testing. In some methodologies development, testing and implementation are stand-alone phases. Once the software has been developed and tested, only then it can be deployed to production. There are still possibilities, however, that when software is implemented, errors could occur. When those errors have occurred, it means that the production environment malfunctions and the operational specialists have to roll back in order to repair the production environment.

Some organisations are still conducting their business manually, rather than automating their processes and activities with software. Some may be operating with existing software which needs replacement. New software could be implemented by one of the various approaches, namely pilot, parallel or big bang (Okrent & Vokurka, 2004). Hertzum, Bansler, Havn and Simonsen (2012:2) explain a pilot implementation "as a field test of a properly engineered, yet unfinished system in its intended environment, using real data, and aiming; through real-use experience; to

explore the value of the system, improve or assess its design and reduce implementation risk”.

Pilot implementation makes it possible for the software to be tried in one area and, when found acceptable, can be rolled out in other areas. Alternatively, *big bang implementation* refers to replacing the existing software with a new one at one go (Capaldo & Rippa, 2009). The risk with big bang implementation is that should software go awry, the organisation must quickly roll back in fast efforts to make the production environment functional again. *Parallel implementation* refers to when the old software runs parallel with the new software to ensure data integrity as well as data migration (Okrent & Vokurka, 2004). With this strategy, both old and new software runs parallel for a particular period of time and when the new software proves stable, the old one is switched off. However, prior to software implementation, regardless of the implementation approach, software testing must occur to verify that the software functions as expected.

2.5 Software testing

Software testing occurs throughout the development or enhancement of existing software. The intention of testing software is to find defects, mistakes or missing requirements in the software under test (Jamil *et al.*, 2016). This is imperative to deliver quality software that will enable the organisation to do business smoothly and remain competitive. The software under test is verified against the business requirements to detect any mistakes that might have been committed by software developers. As explained, software can be conducted manually or automatically through the use of software testing tools.

Manual testing requires the software tester to play the role of end user by performing features on the software under test to ensure its expected behaviour (Bamotra & Randhawa, 2017). Automation testing is performed through software testing tools to reduce the necessity for manual or human participation and repetitive tasks (Singla & Kaur, 2014). According to Hooda and Chhillar (2015), software testing tools are mainly used to conduct performance testing because of the difficulty of testing load manually.

Automation testing is designed to re-run test scenarios, previously performed manually, quickly and repetitively (Waje *et al.*, 2014). Moreover, it increases the depth and scope of testing to improve software quality. Automation testing assists in

executing many complex test cases during every test run supplying coverage that is impossible with manual tests. Performance testing, to assist in testing non-functional requirements of the software, is conducted to determine inadequate performance behaviours of the software under test such as longer execution time and/or lower throughput (Luo, 2016). According to Bhatti and Kumari (2015), there are several types of performance tests – load, stress, volume, endurance, spike and scalability testing – which can be executed to measure performance.

These types of tests allow the performance testers to test the non-functional requirements of the software under test. The Department of Education in Gauteng as well as Safair Airlines experienced performance problems after implementing their software: “The Minister of Education in Gauteng, Panyasa Lesufi, said the software crashed after it received 600 hits per second. The software was upgraded to receive 3 000 hits a second but that still failed and was increased again to 20 000 hits a second” (Monama, Ndlazi & Mabotja, 2016:n). “The airline experienced high traffic volumes in sales due to its R1 birthday bargain on flight tickets. The company says it managed to sell at least 5,000 tickets so far and will extend the deadline for the sale” (Koza, 2015:n). But having tested the functional requirements of the software does not necessarily mean the entire software has been tested. There are also non-functional requirements that require testing as well.

It is much less expensive to correct defects sooner rather than later in the software development life cycle. Some goals of software testing, then, are to identify the correctness, completeness, security and quality of developed computer software, thereby determining the status of the product during and after the build (Khojasteh, Zeki, Naji & Sanatnama, 2012). According to Kapur, Yadavalli and Kumar (2006), software engineering (body of knowledge) does not only help to deliver functional software on time and within the budget, but aids in satisfying specific quality standards. There are industry quality standards that must be adhered to in regard to product quality for the software to be globally accepted.

ISO/IEC/IEEE 29119 is the latest international standard intended to be more inclusive by encompassing various levels of testing processes (Alaqail & Ahmed, 2018). It covers an internationally agreed set of software testing standards readily adopted by any organisation for conducting any type of software testing (Matalonga, Rodrigues & Travassos, 2015). These standards place emphasis on some software testing aspects such as concepts and definitions, testing processes, testing

documentation, test design techniques and keyword-driven tests (Alaqail & Ahmed, 2018). Organisations adopting these software testing standards are able to compete globally and locally because they are exposed to international agreed standards. The table below presents the main types of functional testing that can be conducted:

Table 2.1: Software testing levels (Hooda & Chhillar, 2015)

Unit/Component Testing	It is performed by software developers to ensure that the units of the software in isolation work as specified in the requirement specification.
Integration Testing	Testing the communication and interaction between different modules to ensure that data flows correctly between various components.
System Testing	Testing the entire system to ensure it functions as stipulated in the software requirements specification.
User Acceptance Testing	Testing the final software with the client to ensure it accomplishes intended functionality.

However, to successfully achieve the above-mentioned testing levels, the software testing must adhere to the software testing process. It is vital for the team to know what type of testing to conduct, how and when to conduct it.

2.6 Software testing process

As much as planning is critical for software development, the same applies to software testing. The software testing process is required early in the life cycle, prior to any system coding and during each of the stages preceding implementation (Munassar & Govardhan, 2010). According to Skidmore (2006), methodologies such as V-Model provide a relationship between development and testing in ensuring proper testing and quality assurance throughout the entire project life cycle, as illustrated in figure 2.1 below:

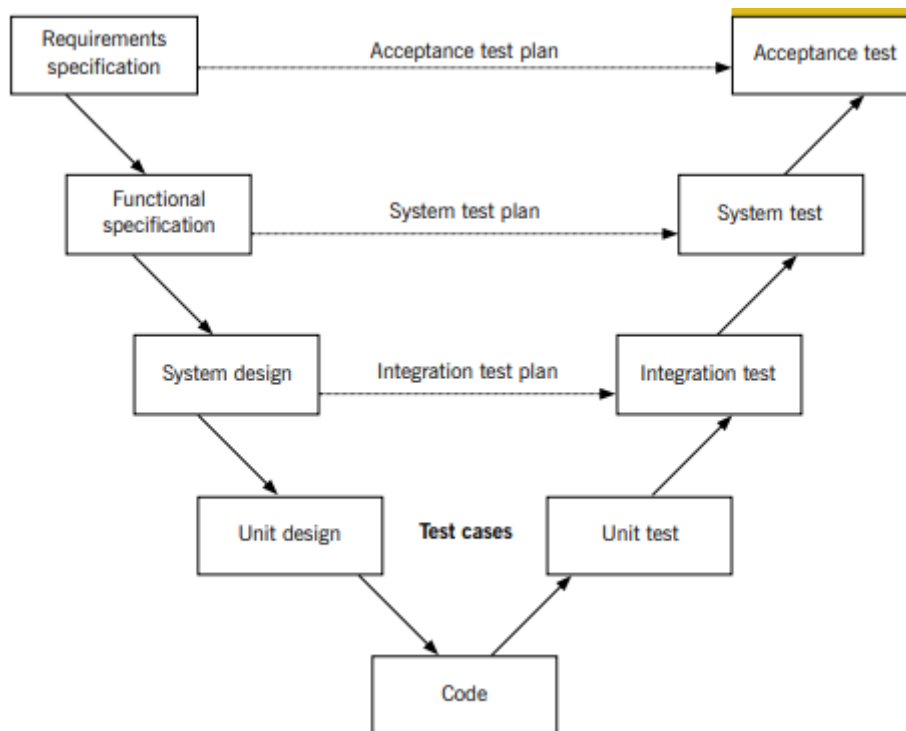


Figure 2.1: V-Model (Adapted from Skidmore, 2006)

Kasurinen (2012) asserted that a test process encompasses test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria, reporting and test closure activities.

The software process offers the flow of the software and expands the assurance of the software product under production (Hooda & Chhillar, 2015). Test planning begins with the creation of the test plan. This first step, the test plan, ensures that the testing activities are adhered to and determines precisely what the testing is meant to achieve. The test plan specifies the items to be tested, the level of testing, the sequence of testing, the manner in which the test strategy will be applied to the testing of each item, as well as description of the test environment (Agarwal, Sharma & Nikhil, 2012). With such details, the test plan establishes a clear indication to stakeholders pertaining to the software testing.

The test process describes the test analysis and design as the activity of designing the test cases using the techniques selected during planning. This can be successfully achieved if the software tester understands the user requirements. However, if the software tester fails to understand the user requirements and architecture software under test, it would not be possible to create test cases which will reveal more errors in short amount of time (Quadri & Farooq, 2010). A test case

outlines the steps required to test any functionality of the software and contains expected and actual result (Hooda & Chhillar, 2015). Such a comparison means the software tester can readily determine whether or not the software under test satisfies requirements or works correctly. These test cases could be captured in a spreadsheet or a testing tool if the organisations have one.

2.7 Software testing methods

Software testing methods are basically the approaches that can be adopted to test and evaluate software within the organisation. It points out which direction to follow when conducting software testing. According to Mishra, Ostrovska and Hacaloglu (2017), the two commonly used types of testing methods are black box (functional) and white box (structural) testing. Hussain and Singh (2015) also refer to black box testing as behavioural testing whereby the software is tested without the knowledge of the internal workings of the software. White box testing is where the software tester knows the internal workings of the software (Jamil *et al.*, 2016). Nidhra and Dondeti (2012), asserted that in white box testing, the software tester is granted access to the code when performing white box testing to test the code of the software.

There is also a third testing method that has more recently been introduced: grey box testing. In grey box testing, the software tester has limited knowledge about the internal workings of the software (Jan *et al.*, 2016). Combining the strengths of both white box and black box testing (Khan & Khan, 2012), with this kind of testing the software tester is not required to have full access of the software's source code. These testing methods can be used in conjunction with various software testing tools.

2.7.1 Black box method

Black box method is concerned with examining the functionality of the software without looking into its internal workings of the software. Black box testing is performed to compare the actual functionality of the software with the intended functionality described in the software specification document (Ahamed, 2009). Mainly conducted to test the behaviour of the software, this method is divided into various techniques which includes equivalence partitioning, boundary value analysis testing and decision table testing (Williams, 2006). Other black box testing techniques are the cause-effect graphing techniques, comparison testing, fuzz testing and model-based testing (Irena, 2008). Table 2.2 describes the various black box techniques:

Table 2.2: Black box techniques (Hussain & Singh, 2015)

Equivalence Partitioning	This technique divides the input domain of a program into equivalence classes, a set of valid or invalid states for input conditions.
Boundary Value Analysis Testing	It is a positive or negative test focused on boundary or limit conditions of the software being tested.
Decision Table Testing	It is a test designed to execute the combinations of inputs based on conditions shown in the decision table.
Comparison Testing	It is a technique whereby the software engineering teams produce independent versions of the system and each version is tested with the same test data, so the same output can be confirmed.
Fuzz Testing	It is a degree to which a software can function correctly in the presence of invalid inputs.
Model-Based Testing	It is an automatic generation of efficient test procedures using models of system requirements and specified functionality.

During black box testing, some parts of the software, especially the back end, are not tested at all (Mishra & Pradhan, 2012), allowing the testing team to actually perform tests on only a selected number of test scenarios which leads to limited coverage (Khan & Khan, 2012) as a result of the lack of knowledge of internal workings of the software. Aichernig (2001) contends that the black box approach does not consider how the test-object is implemented but considers what its requirements are. Some important parts of the tested software may be easily overlooked. The alternative to the black box testing is an approach known as the white box testing method.

2.7.2 White box method

The *white box method* is concerned with testing software by examining the internal workings of the software. It requires the tester to possess the internal knowledge of the software as well as the programming skills. It is typically effective in validating design, decision, assumptions and finding programming errors and implementation errors in the software (Khan, 2011). The testing is based on the code coverage, paths, branches and conditions. The intention is not to find every software defect that exists but to expose situations that could negatively impact the customer. This method is divided into various techniques: control flow/coverage testing, basic path testing, loop testing and data flow testing (Nidhra & Dondeti, 2012). Table 2.3 below describes the above-mentioned white box techniques:

Table 2.3: White box techniques (Khan & Khan, 2012)

Control Flow/Coverage Testing	It uses the flow of the program as a model to control flow and favours simpler paths over less but complicated paths.
Basic Path Testing	It ensures that each possible outcome from the condition is tested at least once.
Loop Testing	It exclusively focuses on the validity of loop construct.
Data Flow Testing	It ensures that the control flow graph has the information about how the program variables are defined and used.
Branch Testing	It ensures that every option (true or false) is tested on every control statement, including compound decisions.

It is wise to begin testing early in the system development life cycle. Steegmans *et al.* (2004) state that the biggest limitation of white box testing is that test suits can only be developed late in the life cycle of a software component. And beginning such testing late in the development life cycle affects testing timelines negatively as it leads to lapsing deadlines. As a result, even more time is required, increases the testing costs. There are, however, software testing tools in place that save time and cost. White box requires intimate knowledge of a target system, testing tools and coding languages and modelling (Acharya & Pandya, 2008). These testing methods can be used in conjunction with various software testing tools. Another testing method that can be used is the grey box method.

2.7.3 Grey box method

The *grey box method*, a combination of black box and white box testing, is a method used to test the software with limited knowledge of the internal workings of the system (Sawant, Bari & Chawan, 2012). Moreover, it is the testing approach used when some knowledge of internal structure is known, but not in detail. Saxena and Singh (2014) argue that the purpose of grey box testing is to examine if there is any defect due to improper structure or usage of the software. According to Bhasin and Kumar (2015), this method is well suited for web applications, web services, functional or business domain testing, security assessment, GUI and distributed environments. Below, Table 2.4 describes the above-mentioned grey box techniques:

Table 2.4: Grey box techniques (Acharya & Pandya, 2012)

Matrix Testing	The software developer begins by defining all the variables that exist in their programs and each variable has an inherent technical risk.
Regression Testing	This testing is performed after making a functional improvement or repair to the program to ensure that what was fixed have not affected other aspects of the program.
Pattern Testing	It helps to dig within the code and determines why the failure has happened.
Orthogonal Array Testing	It is a statistical testing technique that is extremely valuable for testing complex applications.

Some parts of the software may be missed due to limited access to internal workings of the software, resulting in partial code coverage (Saxena & Singh, 2014). This, however, defeats the purpose of end-to-end testing. The aim of software testing is to cover most parts of the software as much as possible. Due to restricted knowledge of the tester, it is not feasible to cover every part of the software with the situation that many program paths go untested. According to Archarya and Pandya (2012), grey box testing continues to rely on how well the software throws exceptions and how well these exceptions are spread within a distributed web service environment. The tester relies on how the software reacts. These testing methods can be used in conjunction with various testing tools.

2.8 Software testing tools

The software automation tools enable software testers to create scripts that can run automatically to test the software. According to Hoffman (1999), software automation is valuable to enhance the tester by performing tasks that are tedious if not impossible for a human or are more cost effective to automate. Automation testing is when the tester writes scripts for testing the software. Such scripts, running over and over again at no additional cost, are much faster than manual tests, capable of reducing the time to run repetitive tests from days to hours. Kaur and Kumari (2011) argue that manual testing is time consuming, resource intensive and allow some defects to remain uncovered. Therefore, automation testing tools are there to help uncovered defects.

Automated testing diminishes the cost of producing software while simultaneously increasing its reliability (Shao, Khurshid & Perry, 2007). Due to the complexity and increasing size of software, testing efforts are expected to increase; therefore, automation testing arises as a practical necessity to reduce time and cost (Mandi & Kumar, 2013). Furthermore, automation testing reduces the amount of manual work, increasing high coverage by executing more test cases and eliminating human errors especially when people tire after multiple repetitions (Nawaz & Malik, 2008). Automation testing will also likely rectify some of the other problems, but it certainly is not a panacea for solving all problems (Tretmans, 1999). In fact, it is always best to possess the skills for utilising automation tools because without these skills, it is pointless to possess such tools.

Software automation tools can be either proprietary or free open source. The proprietary software testing tools are commercialised and require licensing per use. Free open source tools, on the other hand, are free and downloadable from the Internet. According to Singh and Tarika (2014), the free open source automation software testing tools do not require licences to be purchased for use and the software code is available to the user for further enhancements. Some open source automation tools include Apache Selenium, Geb, Windmill, GitHub Protractor, SpecFlow, Tyto Software Sahi and BSDW (Saravanan & Prasad, 2016). Commercial automation testing tools include HP Unified Functional Testing, IBM Rational Functional Tester, Oracle Application Testing Suite, Borland Micro Focus SilkTest, SmartBear Test Complete and Testing Anywhere (Waje *et al.*, 2014).

Also, there are performance software testing tools which can be both open source and proprietary. According to Abbas *et al.* (2017) currently there are various open source and commercial load testing tools available on the market such as LoadRunner, Apache JMeter, LoadRunner, Siege and Microsoft Visual Studio (TFS). Therefore, organisations have variety of software testing tools to choose from. The table below illustrates the functions of various automation tools:

Table 2.5: Software Testing Tools (Sharmila & Ramadevi, 2014)

Apache JMeter	It can be used to simulate a heavy load on a server, network or object to test its strength or to analyse overall performance under different load types.
NeoLoad	It is used for measuring and analysing the performance of the website.
LoadRunner	It is quite useful in understanding and determining the performance and outcome of the system when there is actual load.
LoadUI	It is open source and load testing software used for measuring the performance of web applications.
WebLOAD	It is a tool used for load testing and stress testing.
WAPT (Web Application Performance Tool)	It is an analysing tool for measuring the performance and output of any web application or web-related interface.
Rational Performance Tester	It is an automated performance testing tool which can be used for a web application or a server-based application where there is a process of input and output involved.
Testing Anywhere	It is an automated testing tool which can be employed for testing the performance of any web sites, web applications or any other objects.

Software testing tools are used as part of the testing phase within the software development lifecycle (SDLC) to automate certain tasks, improve testing efficiency and discover issues that are likely difficult to identify using manual testing alone. The aim of this study, then, is to create a decision support system framework for testing and evaluating software within the organisation.

2.9 Decision support system

Decision support systems (DSSs) are designed to assist individuals or groups with decision making in solving problems. Hertz, Cavalieri, Finke, Duchi and Schönsleben (2014:71) describe DSS as “interactive computer-based systems that help people use computer communications, data, documents, knowledge, and models to solve

problems and make decisions". Decision support systems replace human decision making as they help humans to make informed decisions regarding problems they are facing, thereby enhancing the decision-making process. DSSs are not designed to automate decisions but to fairly support decision making because they are flexible enough to react to changing requirements (Hertz, Cavalieri, Finke, Duchi and Schönsleben, 2013). According to Engel, Choi, Harbor and Pandey (2003), typically most DSSs have three main components: a model system, a data system and user interface.

Each component fulfils its particular activity within the DSS. The data relating to the problem is stored in the knowledge base, the model generates decisions based on the content of the knowledge, and the user interface allows users to build models and attain decision support through the adjustment of input parameters (Hosio, Goncalves, Anagnostopoulos & Kostakos, 2016). DSSs are available to managers in support of decision making processes for solving complex issues (Athanasiadis & Andreopoulou, 2011). Therefore, decision makers can utilise these tools to compile useful information from documents, raw data and personal knowledge to make decisions in solving problems. Athanasiadis and Andreopoulou (2015) further state that some DSSs give structured information directly to managers and store knowledge which is availed to managers anytime it might be needed.

At times, it can be surprisingly difficult for people to make decisions, especially when they do not understand the root cause of the problem. Thus, Filip *et al.* (2017) highlight that in order for the decision-maker to overcome limits and constraints encountered, they need DSS to assist them in making difficult decisions for solving complex problems. Decision-making is one of the essential activities of business management, a huge component of any process of implementation. According to Liu *et al.* (2010) various DSSs were developed to support decision makers at all levels in the organisations, including systems that could support problem structuring, operations, financial management and strategic decision making, even extending to support optimisation and simulation. Even in software testing and evaluation, DSSs can assist managers in making right decisions. For this study, two theories were identified: the actor network theory (ANT) and the Diffusion of Innovation (DOI) theory.

2.10 Underpinning theories

The study aimed to develop a decision support system framework for testing and evaluating software. This research is underpinned by two theories, actor network theory (ANT) and Diffusion of Innovation (DOI), meaning that these theories guide the study from two different perspectives. The theories are discussed below.

2.10.1 Actor network theory

The study aims at developing a decision support system framework for testing and evaluating software within the organisation. In this regard, this study is underpinned by the actor network theory (ANT). This theory, originating from sociology, focuses on bringing people and objects (i.e. technology) together through processes of translation to form heterogeneous networks with similar interests (Cho, Mathiassen & Nilsson, 2008). People and objects are referred to as actors. In ANT, people are not the only entities to act nor beings with agency, all are actors (including objects), playing an equal role within the network (Luoma-Aho & Paloviita, 2010). Each actor has something to contribute for the functionality of the network.

Actor network cannot exist without the actor or the network. Teles and Joia (2011) posit that actor network theory is a combination of agency and structure or context in which none of them (actor-network) exist independently of the other. The two complement each other in way that if one is absent the actor network combination becomes completely dysfunctional. Consequently, it is vital to understand the entire concept of actor network. Williams-Jones and Graham (2003) state that for us to distinguish the origins of power and structure in a network, we need to consider all the components that collaborate, co-operate, compete and lead to creation, persistence or perishing of that network. The actor network proposes that knowledge is created, nevertheless the creation is the result of a heterogeneous/diverse network of people, devices and texts which render a form of steadiness (Steen, 2010).

As with any other theory, the ANT has been criticised for treating human and non-human actors equally. Williams-Jones and Graham (2003) argue that humans have different (superior) moral status from objects. There is a belief that objects cannot operate themselves but rely on humans to make them operational. Therefore, human and non-human actors supposedly *cannot* be treated equally. According to Bruun and Hukkinen (2003), however, actors (human and non-human) are studied differently: humans are interviewed while non-humans are investigated via the

mediation of humans (for instance, research reports). This is therefore regarded as a contradiction in ANT.

Black box methodology, one of the tenants of the actor network theory, is described by Lihosit (2014) as something that is not easy to understand or explain. For instance, while not much is known about new technologies, people still eagerly adopt these technologies, assuming they can do what they want. Therefore, this new technology could be regarded as a black box. With black box, while inputs and outputs are both known, the process of arriving at those outputs is often taken for granted (Besel, 2011). The process concerns the internal workings of the black box which is not known. In order to understand the internal workings of the box, it needs to be opened. The black box is opened to gain more insight into the construction of effectiveness data within collaboratives (Broer, Nieboer & Bal, 2010). According to Besel (2011), black boxes consist of knowledge which is accepted and used on a regular basis as a matter of fact. And not only the new technologies, but the actor too can in many ways also be regarded as a black box: when the cover of the box is opened it will constitute a whole network of other, perhaps more complex associations (Tatnall & Gilding, 1999).

Both ANT and software testing consists of black box testing. In ANT, black box is something that is not easy to understand or explain. In software testing, black box is when the software under test is tested without regards for the internal code structure, implementation details and knowledge of internal paths of the software (Acharya & Pandya, 2012). The similarity between the black box within ANT and software testing is that the internal workings of the software or technology are unknown. However, in ANT, this black box needs to be open to gain understanding, whereas in software testing there is no need to open the black box because the purpose is to test the functionality of the software, so it is not important to know the internal workings of the software but rather to ensure that the software behaves as expected.

Lee and Oh (2006) posit that ANT is a theory that helps analyse the ways in which actors form coalitions and involve other actors within the network to strengthen such coalitions and to secure its interests through the use of technology. Through the lens of ANT, the researcher intends to examine the testing and evaluation of software to create a decision support system framework. The lens of ANT involves four stages of translation: problematization, interessement, Enrolment and mobilization (Chen,

Zhang, Zheng & Ciu, 2009). These stages are referred to as the *moments of translation*.

2.10.2 Moments of translation

In ANT, the translation is triggered by the four moments of translation. The term *translation* is associated with the network in terms of representation of actors or networks. *Translation* is described as a way of collaborating different entities and convincing them to have interest in connecting and relating to produce results (Van Der Duim, 2007). The translation occurs between humans and objects once the actor-network has been formed (Comber, Fisher & Wadsworth, 2003). The translation begins with problematization, whereby the issues or problems are defined with relevant actors with the intent of resolving them through the obligatory passage points (Potts, 2009). Obligatory passage point (OPP) is a channel through which all the actors have to pass to satisfy the interest endorsed by the focal actor (Timpka, Bang, Delbanco & Walker, 2007). The primary/focal actor aims at becoming OPP for the network (Luoma-aho & Paloviita, 2010). The focal actor becomes indispensable or irreplaceable within the network. The roles to be played by actors within the actor network are also identified during problematization.

It is during the second stage, *interessement*, where the focal actor persuades, motivates and negotiates with the actors to get them interested and involved in the network (Luoma-Aho & Paloviita, 2010). Actors are not forced to participate in the actor network but are given a choice to do so. It is through *interessement* where the focal actor attempts to impose and stabilise the identity of other actors in the same network (Lee & Oh, 2006). The third stage is *Enrolment*, whereby the actors accept the roles defined for them when enrolling in the network (Iyamu & Roode, 2010). The alliance of networks is formed with the aim of creating an agreement between the stakeholders regarding their interests (Alcouffe, Berland & Levant, 2008).

The focal actor ensures that any kind of training required by the actors is offered for the network to be productive. *Mobilisation* is the final stage in which the focal actor uses a set of methods to ensure that all actors have spokespersons to represent other actors to avoid betrayal by various collectives (Gunawong & Gao, 2010). At this stage, the formed network begins to operate at new targets to implement the proposed solution (Van Der Duim & Van Marwijk, 2006). However, the actors become productive and efficient when they know they are well-presented by their spokespersons; a sense of security enables the actor network to reach targets.

Within the moments of translation, the focal establishes itself indispensable and sets the obligatory point of passage (OPP), the channel through which the actors should pass (Lee & Oh, 2006). Iyamu and Sekgweleo (2013) further describe OPP as an entity that is liable for representing other actors in a way that suits their significance and actions in the world of translation. OPP is a state of little or no negotiation (Tatnall, 2014). Actor network theory consists of key conceptual tools which provide descriptions about the terms used in the network. Table 2.6 below describes the key conceptual ANT tools:

Table 2.6: ANT Tools (Iyamu & Sekgweleo, 2013)

Tenets	Description
Actor (or Actant)	Both human beings and non-human actors such as technological artefacts.
Actor Network	Heterogeneous network of aligned interests, including, for example, people, organisations and standards
Enrolment and Translation	Creating a body of allies, human and non-human, through a process of translating their interests to be aligned with the actor network
Delegates and Inscription	Delegates are actors who “stand in and speak for” particular viewpoints that have been inscribed in them
Irreversibility	The degree to which it is subsequently impossible to go back to a point where alternative possibilities exist
Black Box	A frozen network element, often with properties of irreversibility
Immutable Mobile	Network element with strong properties of irreversibility and effects that transcend time and space

The actor-network suggests that knowledge is created, though the creation is the result of a heterogeneous/diverse network of people, devices and texts which form a balance (Steen, 2010). ANT will be used as a lens to zoom into actors involved in the evaluation and testing of software within the organisation.

2.10.3 ANT and information systems

ANT is a socio-technical theory which could be applied in various disciplines such as organisational studies, accounting, science and technology and economic sociology. Hanseth, Aanestad and Berg (2004) contend that ANT accepts that networks are socio-technical and assist in better understanding the relationship between the social

and the technical system. Information system (IS) is therefore a combination of technical and non-technical resources intended to support various requirements of business within the organisation (Iyamu & Sekgweleo, 2013). As a result, IS is vital as it enables an organisation to operate efficiently and effectively and to remain competitive.

The formation or existence of IS is achieved through the collaboration of a heterogeneous network. Tatnall (2014) state that equal contribution of both human and non-human actors within the network makes the existence of IS a reality. Actors work together to deliver the requested information system by the organisation (Dwivedi, Henriksen, Wastell & De, 2013). Iyamu and Sekgweleo (2013) view ANT as a network formed by various elements such as humans, technological artefacts, organisations and institutions. It takes a collective to create information systems, a collective such as human actors (e.g. project managers, business analysts, software developers, software testers and those who implement it) as well as the technology (hardware and software) required to deliver the information system.

2.10.4 Diffusion of innovation

The diffusion of innovation (DOI) theory was also employed in the data analysis of this study. DOI theory was introduced by Rogers in 1962. He described DOI as the process by which the innovation is communicated to the members of social system via certain communication channels to multitudes over time (Chang, 2010). This theory is concerned with introducing new ideas or technologies to the target market, and with this new technology being introduced to the organisation, it is evident that resistance from systems users usually occurs. Systems users may not like the new system (because it has new features, is difficult to use, requires odious user manuals) and prefer the older one (because there is nothing new to learn, they know it by heart) or they just do not want to change. Hence, Rogers identified five significant characteristics of the innovation that influence its adoption: relative advantage, compatibility, complexity, trialability and observability (Zhai, 2011).

Relative advantage describes how potential adopters expect the innovation to improve their lives (Montfort *et al.*, 2009). This new concept is expected to bring change and simplify how things are done within the organisation. *Compatibility* refers to the degree to which an innovation is perceived as being consistent with the existing values, past experiences and needs of potential adopters (Alemneh & Hastings, 2010). *Complexity* is a point to which an innovation is alleged as difficult to

understand and use (Raus, Flügge & Boutellier, 2008). New technologies or concepts may be difficult to apply because individuals must learn how to use them and must get used to them. *Trialability* refers to a point at which an innovation technology may be tested on a limited basis before adopting (or rejecting) with a decision (Soroka & Jacovi, 2004). *Observability* is the degree to which the results of an innovation are visible to others (Luqman, Abdullah & Ghapar, 2011).

Innovation is the new idea that is developed to be adopted by the social system. The rate of adoption is measured according to how it is accepted within the social system. According to Alqahtani and Wamba (2012), DOI theory investigates how, why and at what rate, new ideas or technologies spread through cultures. Culture is the way in which people live, behave and do things within a particular environment. As a result, the key elements in diffusion of innovation come into play. Montfort *et al.* (2009) describe diffusion as the procedure by which innovation is transferred through certain channels over time among the social system. Below is a diagram portraying various tenets of DOI.

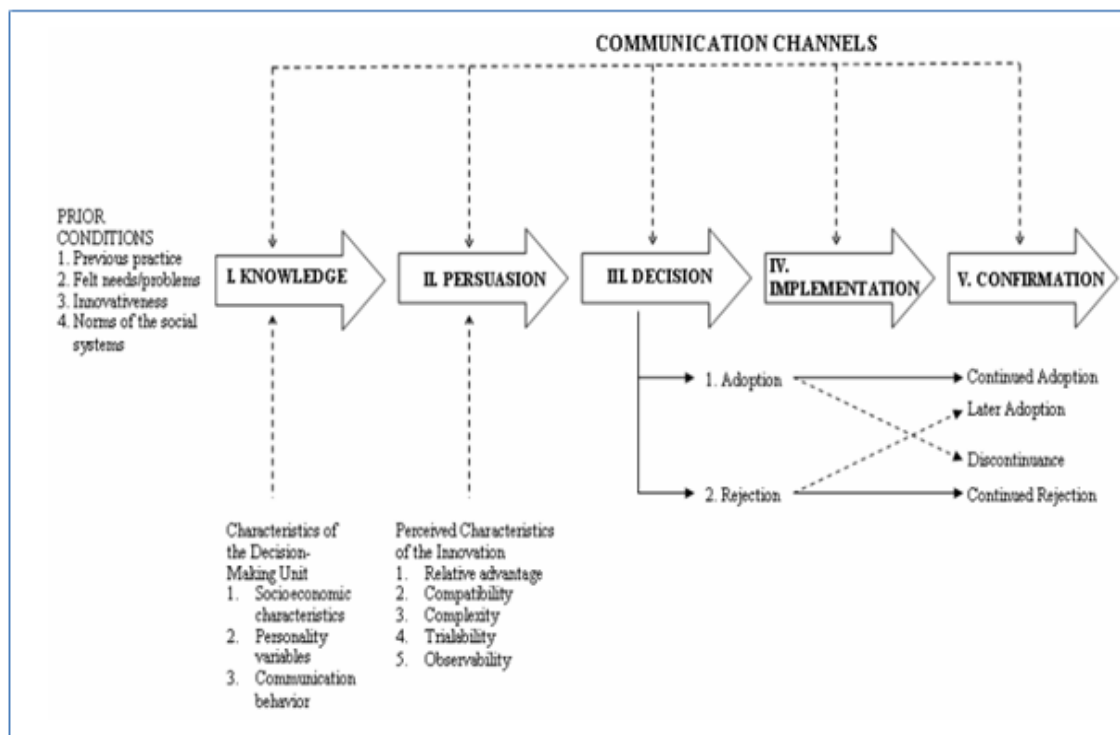


Figure 2. 2: Innovation-decision process (Adapted from Nemutanzhela & Iyamu, 2011)

2.10.5 Innovation-decision process

Diffusion, occurring through the innovation-decision process, consists of five stages that can be followed to diffuse the innovation to the social system: knowledge, persuasion, decision, implementation and confirmation. These stages are described in the table below:

Table 2.7: Innovation-decision process (Sang & Tsai, 2009)

Knowledge	The earliest awareness a potential user has and the understanding of how the innovation operates.
Persuasion	This occurs once the potential user forms an opinion about the innovation.
Decision	This happens when the user moves toward making the choice to either adopt or reject the innovation.
Implementation	This occurs when the user begins to use the innovation.
Confirmation	This will occur when the user seeks reinforcement of the decision to use the innovation.

Innovation is not an overnight occurrence, but rather occurs over a period of time. Therefore, the above-mentioned process plays a vital role because the target market needs must be informed about new ideas and technology to decide whether to accept it or not. Technology incessantly changes; therefore, it is vital to keep up with all the changes. Those who reject technological changes are 'left behind' so to speak, and those who accept it, reaps the benefits associated with new technologies.

2.10.6 DOI and information studies

An *information study* is the merger of library science and information science. Library science is concerned with the practices, perspectives and tools of management, information technology, education and other areas to libraries. The information science is concerned with how information is gathered, analysed, organised, manipulated, stored and retrieved. Tumuhairwe (2013:2) defines *information studies* as "education entailing librarianship, information management, records management and archive practice and teacher librarianship". Diffusion of innovation is a process of spreading the innovation to the society. According to Overhage and Schlauderer (2012), DOI theory clarifies the *why* and *at what rate* innovations gets diffused to a social system over a period of time.

Information has to be arranged and presented in a way that makes sense to the audience. If the information is not presented well, this obviously causes confusion. Therefore, when diffusing new ideas and technologies to the social system, it is important to do this in such a way that the community understand the pros and cons of that particular technology. Decisions will depend on how well the innovation is understood. Therefore, innovative ideas need to be diffused to inform the social system to constantly adjust to the advantages of technology and estimate the risk of absorbing diffusion information and to ultimately decide on the value of specific technology (Yan, 2009). Without information, it is difficult for the social system to understand the innovation. Communication occurs through the innovation-decision process of DOI, has been discussed above, as this is what helps spread the innovation to the social system.

2.10.7 ANT and DOI

Actor network theory (ANT) has been criticised for treating human and non-human actors equally as it is argued that humans have a different moral status than machines or corporations (Williams-Jones & Graham, 2003) and therefore they require special treatment as compared to objects. The core criticism about ANT is that it is too descriptive and fails to suggest in detail how actors should be viewed, and their actions analysed and interpreted (Cresswell, Worth & Sheikh, 2010). According to Greenhalgh, Potts, Wong, Bark and Swinglehurst (2009), ANT can be best used in conjunction with other theories, particularly in relation to the analysis and interpretation of data (Greenhalgh *et al.*, 2009).

According to Muller (2015), ANT provides concrete theoretical and methodological apparatus which could be applied to the empirical work, especially with terms like 'centre of calculation', 'oligopticon', 'black box', 'immutable mobiles' and 'translation' which help make sense of the formation of associations. Cresswell *et al.* (2010:3) asserted that "ANT helps to conceptualise how different realities are experienced and enacted by different actors, resulting in a more nuanced picture of the dynamic relationships between different actors without neglecting their inter-relatedness". Both human and non-human actors can be part of multiple networks. Steen (2010) explained that heterogeneity of networks of people, devices and texts makes the network steady.

The testing and evaluation of software is considered a challenge, a daunting and complex task in many organisations as it depends upon a diversified number of

complementary skills, processes and tools to thoroughly test and evaluate software. Since actor network theory supports the involvement of both human and non-human actors, it was essential to employ this when examining the testing and evaluation of software. During software testing and evaluation, different networks were formed to carry out the tasks. The application of actor network theory does not only focus on the creation of the network, but also help identify the roles, technologies and the connection between the two, both human and non-human actors, within the network. Due to the close interdependency between the actors, and the influential nature of some actors, it was vital to consider the employment of OPP at all times.

However, with ANT, the innovation that has been created by the network could not be diffused within the organisation. DOI, a complementary theory, seeks to explain the how, why and at what rate innovation is spread within the social community. As the tested and evaluated software had to be diffused within the organisation, DOI was employed to complement ANT. Furthermore, DOI assisted in understanding how decisions were made during software testing and evaluation. Essential decisions had to be made to test and evaluate software successfully. DOI was utilised in this research to understand the decisions made in software testing and how software innovation was diffused within the organisation.

2.11 Summary

This chapter aimed at assisting the researcher in gaining a thorough understanding about testing and evaluating software. It assisted in identifying potential areas of research, the knowledge gaps that require further investigation as well as similar work conducted pertaining to software testing and evaluation. The testing and evaluation of software is a challenging and complex task to perform within many organisations as it requires skilful software testers who understand the software testing processes, standards, procedures and complementary tools to best test and precisely evaluate software. Actor network theory was employed because it supports the involvement of both human and non-human actors for the testing and evaluation of software. During software testing and evaluation, different networks were formed to carry out the tasks. Diffusion of innovation was used because it complemented ANT in terms of making crucial decisions during software testing as well as diffusing the tested and evaluated software within the organisation. Software cannot be left hanging once it has been tested, it has to be diffused in its environment for use. If not it may turn into turn into a white elephant.

No organisation intentionally plans to deploy poor quality software, but there are real possibilities of failure if things are not done properly and established steps not followed. Success relies on how well the processes and activities are planned and executed. Also, support is required from top-level management to the users for the successful adoption of new software. As mentioned earlier, software is not only implemented merely for the sake of having it in the organisation, but to solve specific problems and improve the overall competitive advantage of the organisation. Organisations are now exposed to global competition, so for them to survive, they need to be technologically advanced. The next chapter covers the research methodology that was applied in this study.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter discusses the research methodology that was employed in the study. The research methodology consists of different approaches, methods and techniques carefully applied to achieve the objectives of the study. This present study aimed at investigating for better understanding the testing and evaluation of software within organisations. This chapter is divided into nine main sections. The first section discusses the philosophical assumptions. The second section covers the research paradigm taxonomies. The third, fourth and fifth sections explain the research approach, research methods and research design. The sixth section provides detailed information on data collection. The seventh section describes the data analysis. The eighth section explains the ethical considerations. And finally, the last section concludes the chapter.

3.2 Philosophical assumption

In information systems studies, two philosophical assumptions are common: ontology and epistemology. *Research philosophy* is the belief system and set of assumptions that underpin the creation of knowledge (Biedenbach & Jacobsson, 2016). *Ontology* focuses on the nature of world entities as well as the assumptions of reality about those entities (Ansari, Panhwar & Mahesar, 2016). On the other hand, as Shin (2014) describes, *epistemology* is the study of what humans know, how they know it and how to confirm the knowledge claims.

3.2.1 Epistemology

Epistemology is about what can be known about what exists. Kivunja and Kuyini (2017) argue that epistemology is concerned about the bases of knowledge, its nature, how it is formed, how it can be attained, and how it can be conversed to other people. It provides answers to questions that are asked about what is known. According to Bowleg (2017), epistemology is the study of nature, its scope, and justifies and evaluates the knowledge that is produced. It also enables the researcher to determine the nature of knowledge (whether true or false) through the use of proper methods of evaluation. Levers (2013), contends that epistemological inquiry looks at the relationship between the knower and the knowledge and asks, “how do I know the world?”

In order to achieve quality software, it is important for the software testers to have knowledge of software testing methods and skills to utilise the software testing tools. Therefore, business users would be in a position to use the quality software to perform day-to-day duties as well as rendering services customers. . Henderson (2016) further argues that epistemology emphasises the knowledge needed to resolve a specified research question. In creating knowledge, there are epistemic stances to be followed, such as pragmatic, positivistic, operationalist, referential, instrumental, empiricist, rationalist and realist, which make claims concerning what type of knowledge can be created through research (Tennis, 2008). According to Houghton, Hunter and Meskell (2012), when researchers are selecting a suitable paradigm (ontology, epistemology and methodology), they must ensure that it manifests in research strategies and methods adopted for the research. Similarly, they should be able to achieve the objectives of their research while answering research questions appropriately. The epistemology paradigm was employed for this study as this paradigm guided the researcher in creating knowledge around the testing and evaluation of software.

3.2.2 Ontology

Ontology refers to the underlying assumption made about the nature of reality. Henderson (2016) asserts that ontology is concerned with the nature of social entities and the perception of reality researched. Ontology is applied in different fields of study such as computer science, engineering, mathematics, philosophy and psychology. Busse *et al.* (2015), state that ontology has been used in various disciplines with entirely different meanings. However, “ontology is the most comprehensive of all sciences, insofar as it covers everything that exists” (Busse *et al.*, 2015:31). Irrespective of how different it is defined in all disciplines, it remains the science of being. In fact, Henderson (2016) posits that ontology influences the selection of research objectives, questions and even the methodology.

Software testing is conducted in various organisations. Also, there are various testing methods, techniques and software testing tools. These testing methods, techniques and tools are the same but they are applied differently from one organisation to another. However, these organisations intend to achieve quality software through the use of the testing methods, techniques and tools. Henceforth, Sefotho (2015:30) alluded that “ontology is the starting point of all research as it allows the researcher to start asking philosophical questions about the reality they want to study”. It

questions the assumptions about the way in which the world works and concerns itself with the nature of reality (Ihuah & Eaton, 2013). Moreover, ontology reveals the explanation for the individual about what constitutes fact, as explanation is linked with the question of whether or not social entities need to be observed as objective or subjective.

3.2.3 Methodology

The *methodology* chosen for research is a vehicle enabling the researcher to carry out the specific research based on a topic of choice. It offers a set of methods that can be adopted to investigate a particular case. *Methodology* refers to the research design, methods, approaches and procedures that are utilised in an investigation that is planned for discovering something (Kivunja & Kuyini, 2017). Raadschelders (2011) argues that it is vital for researchers to reflect first on the nature of the study, then pay attention to the research objective and methodology, as finally, embed the research either in ontology and epistemology. They would then be in a position to conduct the research without any confusion.

Carefully considered methodology guides the researcher in the work of the research. According to Saunders, Lewis and Thornhill (2012), methodology can be perceived as a theory, a paradigm of an assumption that builds the foundation for conducting the research. There are a variety of research methods from which a researcher can choose when performing research, including the most popular ones, qualitative and quantitative approaches. The application of these particular methodologies enables the researcher to answer the particular research questions (Henderson, 2016).

3.2.4 Axiology

Axiology is concerned with what the individual values in the particular environment. Thus, researchers are expected to conduct themselves in a particular way. The researcher's values impact how they conduct the research and what they appreciate in their research findings. According to Morgan (2007), axiology does not fall under the philosophy of knowledge (e.g. ontology, epistemology and methodology) but under the philosophy of ethics and aesthetics. As a philosophy that studies judgements about the values (Saunders *et al.*, 2012), it is used to critically examine the diversity of existing questions relating to the crux of values such as good conduct and responsibility (Biedenbach & Jacobsson, 2016). It questions the roles which values play in research choices and emphasises the value judgement capability of the researcher (Ihuah & Eaton, 2013). Clearly, the researcher's values play a vital

role in the outcome of the research. Thus, Saunders *et al.* (2007) suggest that values play an important role in all stages of the research process for obtaining credible research results.

3.2.5 Doxology

Doxology is what is believed to be true. According to Gicheru (2013), the term *doxology* refers to the study of opinion or of what is believed to be true as opposed to epistemology (the study of what *is* knowable). Doxology philosophy does not require science to prove what is believed to be true. It regards the truth as it is. There are additional philosophies besides the ones mentioned above. The research onion, depicted below in Figure 3.1, provides an overview of various philosophical approaches, strategies and choices as well as techniques and procedures for conducting research.

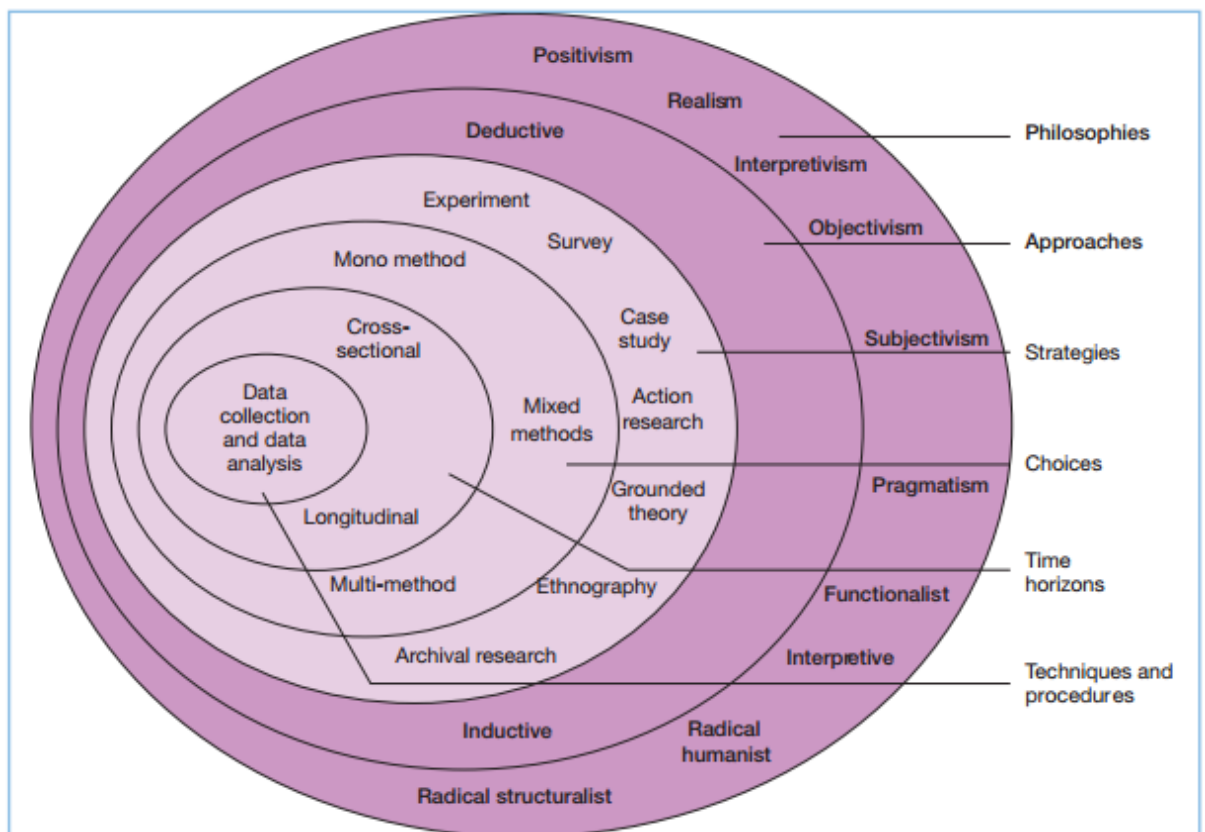


Figure 3.1: The research 'onion' (Adapted from Saunders *et al.*, 2007)

The research paradigms are accompanied by various research approaches in the creation of knowledge. As portrayed in the onion, there are various research paradigms, methods and approaches that can be selected by the researcher depending on what the researcher intends to achieve.

3.3 Paradigm taxonomies

As mentioned earlier, a paradigm is how the world views things. Mackenzie and Knipe (2006) categorise theoretical paradigms as positivist (post-positivist), constructivist, interpretivist, transformative, emancipatory, critical, pragmatism and deconstructivism, postpositivist or interpretivist. These paradigm taxonomies can be embedded in the ontology and epistemology.

3.3.1 Positivism

Positivism is an approach that relies on scientific evidence to reveal the true nature of how society operates. It is of a view that things have to be proven mathematically, with the goal of discovering laws about how the world works in order to create generalisable statements about causal relationships (Schlegel, 2015). Such laws have the status of truth: social objects, then, can be studied in much the same way as natural objects (Crossan, 2016). Positivism is based on the assumption that it is possible to observe social life and establish reliable and valid knowledge about how it works. According to Mack (2010), positivism alleges that all genuine knowledge is based on sense experience and can be advanced by means of observation and experiment. It eradicates a researcher's bias by providing legitimate causality for the research study.

3.3.2 Realism

Realism is the belief that reality lies outside the human mind. Scotland (2012), for example, asserts that realism is the view that objects have an existence independent of the knower. Therefore, realism holds a perception that objects really exist regardless of whether or not they are examined, analysed or studied by science. Kivunja and Kuyini (2017) assert that nature and existence of objects are known to be true by sense experience. For example, the sun rises from the east and sets in the west. The fact is, this cannot change: it is a 'reality'. According to Oppong (2014), as reality exists out there, it is the responsibility of the researcher to access and assess this reality by means of 'objective' data collection techniques. To do so, the researcher uses their subjective or objective mind to verify objects.

3.3.3 Interpretivism

The *interpretive* paradigm enables researchers to explore their world by interpreting what other individuals know or understand. Cresswell (2007) asserts that with interpretive paradigm the researcher aims at making sense or interpreting the meanings other individuals have about the world. Thus, the interpretive paradigm

puts emphasis on recognising and reciting the meaning of human actions and experiences (Levers, 2013). Interpretivists must understand the context of any type of research conducted and the criticality of the interpretation of collected data (Thanh & Thanh, 2015). Consequently, the researcher's interpretation becomes subjective as it is influenced by feelings and emotions.

Yanow and Schwartz-Shea (2011) assert that interpretivist researchers ascertain reality through participants' views coupled with their own background and experiences. Researchers who adopt qualitative research usually collect data from participants through interviews. Finally, that data is analysed to make sense out of it. According to Thanh and Thanh (2015), researchers believe that the interpretivist/constructivist paradigm mainly utilises qualitative methods. Willis (2007:90) emphasises that "interpretivists tend to favour qualitative methods such as case studies and ethnography". Qualitative approaches generally provide rich reports that enable interpretivists to fully understand contexts (Thanh & Thanh, 2015). Thus, interpretivist research is "guided by the researcher's set of beliefs and feelings about the world and how it should be understood and studied" (Terreberry, 2017:55). According to Mack (2010), the role of the interpretivist researcher is to understand, explain and interpret social reality through a different eye. As a result, the researcher's interpretation becomes subjective as it is influenced by feelings and emotions.

The intention of this present research was to create a decision support system framework for testing and evaluating software in organisations. Therefore, this study employed the interpretivism approach to explore participants' subjective views of their experience in their own environments. The researcher subjectively interpreted the qualitative data that was collected for this study.

3.4 Research approach

There are two main research approaches: deductive and inductive. In the inductive approach, the theory is non-existent at the beginning of the research and only evolves as a result of research (Babbie, 2014). The deductive approach constitutes the creation of the assumption grounded on the existing theories, forming a research plan to then test the assumption (Zalaghi & Khazaei, 2016).

3.4.1 Deductive approach

In research, there are two broad methods of reasoning, namely deductive and inductive approaches, which could be used to analyse data. The *deductive approach* enables the researcher to study what other researchers have done, read existing theories of whatever phenomenon is studied and test the hypotheses that arise from those theories. Cho and Lee (2014) assert that the deductive approach enables the researcher to examine existing theory or re-examine the existing data in a new context. It is concerned with testing or confirming hypotheses which leads to confirming or denying the original theory. Therefore, it is important to understand that deductive reasoning (also known as theory-testing) is not just about testing a theory but about refining, improving and extending it (Bhattacharjee, 2012). This approach allows the researcher to apply rules to narrow the facts, moving from general to more specific reasoning, until a conclusion is reached.

3.4.2 Inductive approach

Inductive reasoning works the other way. It begins by detecting patterns and regularities within specific observations. The researcher takes the particular observations and uses them to propose a general theory. The logic moves from specific observations to broader generalisations. Saunders *et al.* (2009) affirm that with inductive reasoning, data is first collected, and the theory is advanced as a result of the data analysis (Saunders *et al.*, 2009). As such, particular instances are observed and combined into a general statement to develop a theory (Elo & Kynga, 2008). This current study aims at developing a decision support system framework to test and evaluate software within organisations. Therefore, the inductive approach will be employed effectively for this study.

3.5 Research methods

There are popular research approaches that can be adopted when conducting research: these include quantitative and qualitative research approaches, as well as a mixed method which combines both qualitative and quantitative (Azorin & Cameron, 2010). A brief discussion of these three types of research methods will follow.

3.5.1 Quantitative methods

Quantitative research methodology is concerned with collecting numeric data in order to describe the phenomena being studied. Yilmaz (2013) describes quantitative research as a type of empirical research that focuses on social phenomenon, testing

a theory that consists of variables which are measured with numbers and analysed with statistics to determine if the theory explains a phenomenon of interest. Quantitative research is compared to the positivistic empirical research which focuses on experimental design and statistical procedures such as multiple regression and structural equation modelling (Petrescu & Lauer, 2017). The data is converted into numerical form through statistical calculations, enabling the researcher to draw a conclusion. This is done with the hope that the numbers will produce an unbiased result that can be generalised to some larger population. The limitation of the quantitative approach, though, is that it does not offer deeper understanding of the social phenomenon due to reliance on simple data sets (Ansari *et al.*, 2016).

3.5.2 Qualitative methods

Qualitative research aims at explaining the social phenomena by examining people's beliefs, culture and experiences. The qualitative approach enables researchers to capture the thoughts and feelings of the research participants, leading to an understanding of the meaning that people ascribe to their experiences (Sutton & Austin, 2015). As it provides rich descriptive accounts of the phenomenon under investigation (Gelo *et al.*, 2008), it is concerned with addressing the social aspects of the world and seeks to find answers regarding people's behaviour, opinions, cultures and differences between social groups (Barker, Linsley & Kane, 2016). The qualitative research method enables the researcher to delve into a deeper understanding of the problem.

This approach investigates the why and how of the subject. Investigations can be conducted on individuals, groups of people, communities, organisations and institutions (Zubber-Skerritt & Fletcher, 2007). Rajasekar *et al.* (2013) characterise this approach as non-numeric, descriptive, applying reasoning and using words to describe the findings. The data collected in qualitative approach tends to be in the form of transcripts, of words rather than numbers. Qualitative research is effective in understanding and explaining complicated situations by obtaining daily knowledge in order to create theories (Petrescu & Lauer, 2017).

The qualitative research approach provides detailed descriptions of how people experience a given research problem (Rajasekar *et al.*, 2013). Researchers obtain such information through the use of various data collection approaches. The qualitative approach consists of the collection, analysis and interpretation of narrative

forms of data (Hayes, Bonner & Douglas, 2013). Such methods are explained in detail under data collection section. The engagement with participants helped the researcher to explore why things occurred, the way they did when testing and evaluating software within the organisation. Therefore, this study employed the qualitative approach.

3.5.3 Mixed methods

Mixed methodology combines the concepts of both quantitative and qualitative research methods. De Lisle (2011) describes mixed methods research as a type of research in which a researcher mixes the elements of qualitative and quantitative research methods for the broad purposes of breadth and depth of understanding and corroboration. Researchers usually adopt mixed methods when their research objectives address both quantitative and qualitative methods within a study. It allows a researcher to gather, examine, combine and pull interpretations from both quantitative and qualitative data within a single study of inquiry (Cameron, 2011), enabling the researcher to answer questions from two perspectives (quantitative and qualitative). The researcher is able to analyse both numeric and non-numeric data in reaching conclusions concerning the subject being examined.

3.6 Research design

Research requires a great deal of information searching and reading based on the objective of the study. Ragab and Arisha (2018:1) define *research* as “systematic investigation into and study of materials and sources in order to establish facts and reach new conclusions”. These facts expand on the existing knowledge. Therefore, a research design is required to guide a researcher in carrying out the research. Wedawatta *et al.* (2011) argue that the research design delineates the complete direction of the research as well as the process by which the research is conducted. The approach selection depends on what the researcher would like to investigate, together with the type of research approach the researcher finds appropriate for the research. Therefore, the research design is not related to any particular research approach for collecting data or any particular type of data.

The research design can be coupled with any research approach. Dube and Pare (2003) describe *research design* as the elements related with the design of the study, such as the nature of research questions, the theoretical foundations as well as the criteria adopted for selecting the cases. The purpose of a research design is to ensure that the evidence obtained enables the researcher to successfully address

the research problem as clearly as possible. Logic also plays a vital role in research as it allows the researcher's work to flow. There are various types of research designs that can be followed, including action research, grounded theory, ethnography, survey and case study.

3.6.1 Action research design

Action research is concerned with learning by doing. Bhattacharjee (2012) defines *action research* as design that accepts intricate social phenomena that is understood when action is introduced to the phenomena and perceiving the effects of such actions. The actions of a researcher have to be based on a theory that explains the why and how such actions may cause the anticipated change (Bhattacharjee, 2012). Thereafter, the researcher observes and learns from those actions to generate insights about the phenomena under study.

3.6.2 Grounded theory

Grounded theory is a systematic research design that enables the researcher to construct theory through data analysis. It provides the capability of analysing data that has been collected using any technique of data collection. Wedawatta *et al.* (2011) suggest that grounded theory is concerned with developing a well-integrated set of concepts that provide a thorough theoretical explanation of phenomena being studied. According to Bryman (2008), grounded theory is created from data that is systematically gathered and analysed through the research process in an iterative process. Therefore, the researcher needs a theoretical framework to analyse the collected data as it assists the researcher in remaining tightly focused and not floundering all over the place.

3.6.3 Ethnography

Ethnography refers to a type of research carried out to study people within their own environment. It is conducted with the intention of gaining insight about people. Hernandez and Fisher (2013) state that ethnography intends to generate knowledge about people and their surroundings, achievable through the use of observation and face-to-face interviews. During observation, the researcher has the opportunity to observe how people do things and is able to document those experiences. Moreover, with face-to-face interviews, the researcher has the opportunity to ask questions and probe for follow up material to elicit more information about the interviewee. According to Wedawatta *et al.* (2011), ethnography necessitates that the researcher become part of the group under study to understand the phenomenon under study.

The researcher's involvement escalates the possibility of obtaining knowledge about the phenomenon being studied.

3.6.4 Survey

A *survey*, a data collection method for collecting data about people, is also used to collect data about things that occur in the real world regarding phenomena under study. Leeuw, Hox and Dillman (2008) explain that a survey can be regarded as method used to systematically collect quantitative data from a large sample extracted from a population relevant to the phenomena under study. It enables the researcher to collect large quantities of data quickly and inexpensively as it consists of questions that help the researcher seek relevant answers to the phenomena being studied. This data can be collected in person, via email, telephonically or online depending on the researcher's choice. Researchers have noted that surveys and experiments have been used primarily in the marketing environment (Petrescu & Lauer, 2017) as it is convenient for marketers to collect necessary information from the target market.

3.6.5 Case study

The *case study* aims at investigating an instance which may be an organisation, department, project, information system, a person or even a kind of illness. Yazan (2015:138) describes *case study* as "a contemporary phenomenon within its real-life context, especially when the boundaries between a phenomenon and context are not clear and the researcher has little control over the phenomenon and context". As it enables the researcher to get clarity to questions such as 'how' and 'why' of an event or phenomena, it is selected for detailed investigations of a particular phenomenon of interest to gain deeper understanding (Nabukenya, 2012). The case study, offering a systematic manner for viewing events, gathering data, evaluating information and reporting results, provides detailed contextual views on phenomenon of interest.

A case can be studied through various means of collecting data such as interviews and questionnaires with the intent of gaining in-depth understanding of the phenomenon. Hilburn, Towhidnejad, Nangia and Shen (2006) posit that a case study involves the application of knowledge and skills, by an individual or group, to the identification and solution of a problem associated with a real-life situation. According to Henderson (2016), a case study is suitable in the environment where there are large numbers of variables in a small number of applied units of analysis when the context is of great importance (Henderson, 2016). Baxter and Jack (2008) highlight that through case study, the problem is not explored through only one lens

but rather with multiple lenses which then reveal several facets of the phenomenon. Yin (2014) argues that a high-quality case study puts emphasis on rigour, validity and reliability.

Three organisations were chosen as case studies to gain a deeper understanding pertaining to how software was tested and evaluated in different organisations. The cases consisted of a private, a public and a small and medium enterprise (SME). Organisations are classified in terms of sector, size, total number of employees, total turnover and total gross asset value. Thus a private company is a privately-owned company that is prohibited from offering its shares to the public and the transferability of its shares are restricted, but it may have more than 50 shareholders (Department of Trade and Industry, 2010). Ownership in a public company is spread among the general public in many shares of stock which are freely traded on a stock exchange but restricts, limits or negates their right of pre-emption (Department of Trade and Industry, 2010).

In South Africa, an SME is any enterprise with fewer than 100 to 200 employees, annual turnover of less than R4 million to R50 million and gross assets, excluding fixed assets less than R2 million to R18 million depending upon the industry, and has direct managerial involvement by owners (Abor & Quartey, 2010). The public organisation has its own way of conducting business which differs from the private organisation and the small medium enterprise (SME). All the three organisations' operational activities are conducted differently. Hence, this helped the researcher to realise how differently or similarly software testing and evaluation was conducted in the chosen organisations.

The way software was tested and evaluated in these organisations varied in numerous ways, including how the software testing was approached, the tools used and the manner in which they were used, why those tools were chosen and which testing methods were applied. The case study, together with the qualitative research approach, was adopted for this study.

3.7 Data collection

After the research design, the next step to consider was data collection. This was achieved by consulting as many relevant participants as possible. Data collection enabled the researcher to systematically collect data about a chosen topic. Data collection methods influenced the test reliability and validity (Yang, Wang & Su,

2006). According to Oates (2006), data can be collected through various data collection methods such as interviews, questionnaires, observations and documentation. This present study employed two different approaches and methods in data collecting process, interviews and documentation.

3.7.1 Observation

Data can be collected in different ways: through observation, experiment, interview and documentation. *Observation* is collecting data by observing how things are done or how they occur. Driscoll (2011) asserts that observation entails observing and measuring the world around us, including observations of people and other measurable events. It is used by researchers to examine people and events in natural settings and naturally occurring situations. As such, what is observed gets documented and will afterwards be analysed.

3.7.2 Experiment

An *experiment* can also be used for data gathering. The purpose of experiment is to study causal links between two variables, independent and dependent (Saunders *et al.*, 2009). An experiment is conducted to support, disprove or validate a hypothesis. Kothari (2004) argues that experiments are conducted to test a hypothesis to discover new relationships. Experiments generate insights into cause and effect by demonstrating what outcomes occur when a particular factor is manipulated.

3.7.3 Interview

Interviews are widely used in research as a data collection tool. Unlike questionnaires, interviews are more powerful in obtaining rich data that allows researchers to examine people's views in greater depth (Alshenqeeti, 2014). An interview, a conversation that occurs between the interviewer and interviewee, has the purpose of enabling the interviewer to understand the meaning of what the interviewees say. Harris and Brown (2010) assert that the intention of an interview is to gather in-depth insights on participant attitudes, thoughts and actions. Interviews enabled the researcher to gather more information from the interviewee regarding the investigated topic. There are three different interview approaches: structured, semi-structured and unstructured (Dicicco-Bloom & Crabtree, 2006).

A *semi-structured interview* is a pre-planned interview where the researcher writes down the interview questions prior to conducting the interview so the interviewer can get the interviewee to elaborate and explain particular issues which are ambiguous

(Alsaawi, 2014). This type of interview is a controlled way of obtaining information from interviews. Therefore, semi-structured interviews were employed for this study as they allow the interviewer to prompt the interviewee for greater clarity on answers that are unclear.

With *structured interviews*, the interviewer does not have to follow the questions as prepared in ascending order but can follow any order depending on the conversation. Consequently, the interviewee is free to respond to questions and even ask questions. This approach allowed the interviewer to take notes during the interview and record the conversation, which then enabled the researcher to focus on the interview content and verbal prompts. This simplifies the interview transcriptions as well (Jamshed, 2014) as by recording the interview, the researcher is able to capture the entire conversation.

Various types of media may be employed during an interview. Such media sources can help to capture the interview conversation and even images. Van Iddekinge, Raymark and Roth (2006) state that rich media sources such video-taping and video conferencing increase credibility because they establish more opportunities to verify information (e.g., by asking follow-up interview questions). Follow-up questions enable the interviewer to gain a heightened understanding of the subject.

The interviewer can decide to take notes during the interview, record the audio of the interview, or video tape it. Whiting (2008) argues that audio recorders, note taking or video cameras are the three most common methods of recording interview data. For this study, the voice recorder and note taking were selected to capture the interview conversation which was subsequently transcribed word-for-word. Interviews were conducted in English as this the common language of the information technology field. Permission to record the conversation was requested prior to the interview. Recording the interview tends to be the best way to retain accurate information because note taking during the interview may cause interviewees to feel as if they are not receiving direct attention; important points might be missed.

3.7.4 Field work

Prior to conducting semi-structured interviews within the organisations involved, the researcher acquired a consent letter from the university for requesting permission to conduct interviews within those organisations. Various organisations were approached by the researcher and permission was granted. In return, the university

expected a signed consent letters from those organisations, which the researcher duly provided. Then, the researcher made appointments with the IT managers of the involved organisations.

Interviews set up

Interviews were conducted at various locations, including manager offices, boardrooms, a cafeteria and even one participant's house. Participants were interview separately, one at a time based on their availability. At Mootledi Logistics, interviews were conducted in the Development and Support Manager's office while others took place in the boardroom. The interviews were conducted in two phases because of time, availability and content. In the first phase, a total of 9 interviews were conducted with the following employees: Project managers, business analysts, software developers, development and support manager, and IT operation manager were interviewed. . From the second phase, additional five interviews were conducted. In total, 14 interviews were conducted when there new information was not forthcoming.

The second organisation where interviews were conducted was Mmuso Technologies. The Test Manager's office was the location for conducting all interviews. The researcher interviewed 14 participants individually, interviewing to a point whereby he was getting the same responses, the researcher decided to stop interviewing because no new information was coming forward. Participants included project managers, business analysts, software developers, software testers and functional support personnel. These participants were interviewed based on their availability.

The third organisation was Bokamoso Solutions and the interviews were conducted at their client's premises. Some interviews were conducted in the boardroom, some the cafeteria, and one at a software developer's house. Initially 10 interviews were conducted, but the researcher felt that more data was required. As a result, two more interviews were conducted and the researcher was receiving the same information as other interviewees. Participants who were available included project managers, software developer/Architects, software testers (test analysts, automation testers and performance testers). They were interviewed individually.

Prior to interviewing participants, the researcher introduced himself and the topic of the study, explaining the code of ethics and requesting the option to record the

conversations. Participants willingly agreed, and semi-structured interviews commenced. During the interview the researcher took notes to probe for clarity on the responses he got from the participants. English was the medium language because it is a common language in the information technology field. At the end of each interview, the researcher conscientiously thanked each participant for taking part in the study.

When the researcher was done with all the interviews for one organisation, the interviews were transcribed word-for-word. Similar transcriptions were also done for the second and third organisation. After transcribing the data, the researcher cleaned the data to be readable. The transcribed data was formatted in Microsoft Word documents in accordance with individual organisations. Pages and lines in each document were numbered. For the purpose of analysis, each participant was labelled. This was to preserve the anonymity and confidentiality of the identities of the participants. Based on these attributes, a referencing standard was formulated and adopted, including the organisation's name, the participant, the page number and the line number. The table below illustrates the total number of participants, from three organisations:

Table 3.1: Participants

Organisation Name	Type of Organisation	Total Number of Participants
Mootledi Logistics	Private Company	14
Mmuso Technologies	Public Company	14
Bokamoso Solutions	Small Medium Enterprise	12

Organisation and participants coding

- **Mootledi Logistics:** participants - *ML01 to ML14*. ML = Organisation name; 01 to 14 indicates the numbers of participants.
- **Mmuso Technologies:** participants - *MT01 to MT14*. MT = Organisation name; 01 to 14 indicates the numbers of participants.
- **Bokamoso Solutions:** participants - *BS01 to BS12*. BS = Organisation name; 01 to 12 indicates the number of participants.

Case study one: mootledi logistics

Mootledi Logistics is located in Kempton Park, in Gauteng. It is a car rental organisation that renders services locally and internationally. It relies on software it

purchases from vendors as well as the software it develops in-house to serve its customers and perform day-to-day duties. However, such software requires testing and evaluating prior to its deployment into production to ensure that business continued as usual without interruptions. Initial interviews and follow-up interviews were conducted at the organisation's premises in Kempton Park.

Interview location

The interviews were conducted in the office of the Software Development and Support Manager. Other interviews were conducted in the boardroom. All the interviews were recorded because the researcher wanted to capture the entire conversations without risking loss of any information. Immediately after the interviewer introduced himself, he then requested permission from each participant to record the interview conversation. Fortunately, all participants agreed to the recording of the conversation. During the interviews, the interviewer was also making notes in preparation for asking follow-up questions. Interviews were conducted up to the point of saturation whereby the interviewer was getting the same responses from the various participants.

Interview duration

The researcher managed to interview 14 participants, including a Development and Support Manager, an IT Operations Manager, project managers, business analysts and software developers. The longest interview lasted for 80 minutes and the shortest interview lasted only 10 minutes and 39 seconds. On the same night after concluding all interviews, the researcher began transcribing those interviews. The interviews were transcribed word-by-word and later data cleaning occurred to convert spoken English into readable English.

Case study two: mmuso technologies

Mmuso Technologies is located at Centurion, Gauteng, with branches in various provinces around South Africa. Mmuso Technologies is responsible for providing government departments with software solutions to fulfil their duties. Once this software is developed, tested and evaluated it is then accepted by the government department that requested it, and adopted to perform day-to-day activities of that particular department. The software is used to render necessary services to the citizens of the Republic of South Africa.

Interview location

Interviews were conducted at the organisation premises in Centurion. The Test Manager's office was used as the destination to interview all participants. All the interviews were recorded because the researcher desired to capture entire conversations without losing any information. However, after the interviewer had introduced himself, he then requested permission from each participant to record the interview. Fortunately, all participants agreed to the recording of the conversation. During the interviews, the interviewer was also making notes in preparation for follow-up questions for additional clarity. Interviews were conducted to the point of saturation whereby the interviewer was getting similar responses from participants.

Interview duration

The total number of participants who were interviewed at Mmuso Technologies was 14. These participants included project managers, business analysts, software developers, software testers and functional support personnel. The same set of questions was asked of all who participated. Thereafter, the interviews were transcribed word-by-word, followed by data cleaning to convert spoken English into writable English. After transcribing the interviews, the interviewer realised that some interviews needed follow-up interviews.

Case study three: bokamoso solutions

Bokamoso Solutions, situated in Illovo, Johannesburg, is an organisation which invests in research to ensure its convergence and relevance to the industry, thereby providing solutions that add value to its market base. Bokamoso Solutions employees are based at client site to offer their specialised services. Those services include software development, infrastructure management, software testing and consulting.

Interview locations

Interviews were conducted at one of Bokamoso Solutions' clients in Johannesburg. Bokamoso Solutions Managing Director liaised and arranged appointments with the manager at the client side who was managing the resources. The manager at the client side arranged a boardroom for the researcher to conduct interviews. The boardroom was used as a destination to interview some participants. Other participants were interviewed at the cafeteria because the boardroom was occupied, while other participants were interviewed at a software developer's house because

they were committed with projects during the interviews. Interviews were recorded because the researcher did not want to risk losing valuable information.

However, after the interviewer introduced himself, he then requested permission from each participant to record the interview. Fortunately, all participants agreed to the recording of the interview. During the interviews, the interviewer was also making notes in order to ask follow-up questions. Interviews were conducted until the point of saturation whereby the interviewer was receiving similar responses from participants.

Interview duration

The researcher managed to interview 12 participants including project coordinators, software developers, and software testers (manual software testers, automation testers and performance testers). The longest interview lasted for 32 minutes and the shortest interview only lasted 6 minutes and 29 seconds. On the same night after conducting all interviews, the researcher started transcribing those interviews.

3.7.5 Documentation

Documentation is the other data collection tool that can be used in research. *Document* is a broad term defined to cover variety of written records, physical traces and visual images (Merriam & Tisdell, 2015). Documents can be classified as either 'found' or 'researcher-generated' documents. According to Kolhe *et al.* (2013), 'found' documents exist prior to research and can be found in most organisations whilst research-generated documents are put together solely for the purpose of research. Access to documents pertaining to policy and strategy related to the study was requested from the organisations and used as supporting documentation for interviews conducted. The researcher was able to get the organisational structures as well as IT structures by employees within the organisation.

On the other hand, 'research-generated documents' include things like interview notes, interview recordings, field notes, answers to questionnaires, journals and articles. Feng and Hannafin (2005) argue that qualitative documentation methods such as tape recordings and written field notes are widely used to collect original data. Multi-method approached research involves combined data collection techniques, such as interviews and documentation organised to provide multiple but different data sets regarding the phenomena (Dube & Pare, 2003). Documentation was also used to supplement data collected from the interviews as this helped to verify the information provided by participants.

3.8 Data analysis

Data analysis occurs after the completion of data collection. Binckman and Rog (2009:234) argue that data analysis is “what does a researcher do with the collected evidence to make sense of it”. However, such evidence requires interpretation to become something that makes sense. Nenty (2009) describes *interpretation* as “analytical thinking that squeezes meaning out of the mere accumulation of facts”. Therefore, a researcher has to apply investigative and analytical skills to the collected evidence and make sense out of it. As a result, the interpretation of data analysis becomes the findings of the study.

Two theories, actor network theory and diffusion of innovation, were used as lenses at both the macro and micro levels in the analysis of the data. ANT was used at macro level, mainly because it guided the DoI, which was used at a micro level to diffuse the innovation. The two theories were employed as follows:

First, ANT was applied to:

- i. understand software testing, through its examination of the problematisation of tools (e.g. for manual, automated and performance testing) that are used for software testing;
- ii. examine the relationship between actors. This was to gain understanding of software testing methods, through their selection and adoption; and
- iii. examine the factors that could influence testing and evaluation of software through moments of translation.

This was followed by the use of DOI, to:

- iv. understand how decisions are made in testing software in organisations; and
- v. explore how software innovation is diffused across the organisations.

The data collected through semi-structured interviews was analysed using ANT and DOI. The data for the first case study was analysed using four moments of translation of ANT, including problematisation, interessement, Enrolment and mobilisation. Once that was completed, the same data was analysed using the innovation-decision process of DOI. They include knowledge, persuasion, decision, implementation and confirmation. The same was done for the second and third cases studies. The

researcher was asking questions such as what, who, how, when, where and why on each stage of the moments of translation as well as on each stage of innovation-decision process. Such questions enabled the researcher to explore relevant answers from the data analysed.

3.8.1 Unit of analysis

A unit is a portion of a complete entity. The unit of analysis is the main entity that is analysed within a study. Bhattacharjee (2012) describes *unit of analysis* as a person, group or object that needs to be explored to be understood better. Elo and Kynga (2007) further describe unit of analysis as a letter, sentence or portion of pages or words, number of participants in discussions or time used for discussion. In this study, three cases were explored, consisting of technical (IT) and non-technical (business) participants. Table 3.2 illustrates the units of analysis.

Table 3.2: Units of analysis

Cases (Three)	Technical (IT)	Non-Technical (Business)
Company A (Private Company)	Project Managers (2)	Business Managers (0)
	Software Developers (5)	Business Analysts (5)
	Software Testers (0)	End Users (0)
	Systems Analysts (0)	
	Support Specialist (0)	
	IT Managers (2)	
Company B (Public Company)	Project Managers (1)	Business Managers (0)
	Software Developers (2)	Business Analysts (1)
	Software Testers (6)	End Users (0)
	Systems Analysts (0)	
	Support Specialist (3)	
	IT Managers (1)	
Company C (Small Medium Enterprise)	Project Managers (2)	Business Managers (0)
	Software Developers (2)	Business Analysts (0)
	Software Testers (8)	End Users (0)
	Systems Analysts (0)	
	Support Specialist (0)	
	IT Managers (0)	

This study makes use of three cases studies which are comprised of both technical and non-technical actors. The technical actors are IT people whilst non-technical actors are business people. Therefore, the data will be collected from both technical (IT) and non-technical (business) actors. Thereafter, that collected data will be analysed accordingly in units. Due to the nature of the study, to test and evaluate software within an organisation, it was deemed fit to interview both technical and

non-technical actors. These participants may have knowledge about what happens within the system development life cycle. The units of analysis enabled the researcher to identify which participants needed to be interviewed.

3.9 Ethical consideration

Ethical consideration is concerned with deciding what information to disclose and how to disclose it. The disclosure of information could occur between two or more individuals. Therefore, the individuals involved have to agree to keep the information a secret prior to its revelation. Revealing such private information could have negative impact on the owner of the information. In research, ethical consideration occurs between the researcher and participants. The researcher has to offer informed consent to those who are participating that whatever they disclose will be private between researcher and participant. *Informed consent* means that a person knowingly, voluntarily and intelligently and in a clear way, offers his agreement (Fouka & Mantzourou, 2011). As a result, the participants are able to disclose information relevant to the research without fear of being discriminated against.

Institutions of higher learning have their own research code of ethics to protect all parties that are involved in a research. Therefore, the researcher had to abide by the CPUT University Research Code of Ethics. The researcher also adhered to each organisation's ethical code. Anonymity was emphasised in the consent form and at the briefings. The researcher provided the consent form and requested that respondents complete it as a guarantee that identities will not be revealed.

3.10 Summary

The aim of this chapter is to present the research methodology for this study. The research methodology serves as a plan, a road map, for how research is conducted. The researcher explained why certain research strategies were chosen for this study. Also, various data collection methods were explained, and reasons provided as to why they were chosen for this study. The researcher also touched on the stages that will be used to analyse data, including the lens of ANT, innovation within DOI as well as the communication channels. The next chapter, which is chapter four presents a detailed information about the three organisations that were used as cases in the research.

CHAPTER FOUR

CASE STUDY OVERVIEW

4.1 Introduction

The research aims to examine the testing and evaluation of software within organisations. Three case studies were employed to understand how software is tested and evaluated within each organisation. The scope of the study was to understand tools used for software testing, understand how software testing methods were selected and adopted, understand factors influencing software testing and evaluation, and understand how decisions were made regarding testing software in organisations, and finally, to explore how software innovation was diffused across the organisations.

This chapter provides an overview of the organisations studied. The same data collection methods and approaches were applied to all cases. The objectives for all organisations were also the same, as the study aims at examining the testing and evaluation of software. However, the organisations were treated as separate cases. The three organisations used as case studies are Mootledi Logistics, Mmuso Technologies and Bokamoso Solutions. All the names used are pseudonyms to protect the identity of the organisations.

The rest of the chapter is structured into three main sections per case. The first section presents the overview, structure and IT structure of the organisation as applied in all three cases: Mootledi Logistics, Mmuso Technologies and Bokamoso Solutions, respectively. The final part is the summary of this chapter.

4.2 Overview: mootledi logistics

Mootledi Logistics is a car rental organisation established in 1967 in South Africa. The Kempton Park branch has 800 employees, 54 of whom are in the IT department. It offers a wide variety of rental cars that best serve the individual or collective requirements. These cars fall within various categories including fleet, vans, trucks, safari vehicles and luxury cars as well as chauffeur-driven services. It is up to the renter to choose what is required. The usual fleet includes a range of small, mid and large cars as well as SUVs. These cars can be used to get around town, attend a conference and to carry the team or just for the pleasure of a luxurious drive. Vans can be used for moving things to new home, transporting materials or to carry heavy

belongings of the renter. The organisation also offers refrigerated trucks and roll-back trucks for the transportation of forklifts and compressors.

Mootledi Logistics safari vehicles have many features it can be compared to an over land version of the Swiss army knife. These vehicles, built for rugged terrain, allow the individual to journey through the wildest landscapes of Southern Africa, making every trip an unforgettable experience. There is also a range of exclusive luxury cars to choose from. Their service is world-class, ensuring the best treatment wherever they go, with personalised meet-and-greet on pickup and delivery of vehicles. Clients choose Mootledi Logistics cars because it's not just about a great rental car, it's about the way the organisation makes clients feel.

All vehicles are less than 12 months old and have fewer than 60 000 kilometres on the clock. Vehicles are regularly serviced as per manufacturer specification and are free of damage, fitted with vehicle management systems to assist Mootledi Logistics luxury car services with real-time geographical position, speed and route monitoring. All vehicles used for Mootledi Logistics luxury car services have valid road transportation permits for the provinces in which they operate. Mootledi Logistics is found in all provinces in South Africa as well as other African countries. However, the particular branch chosen for this study is situated in Gauteng, Kempton Park.

4.2.1 Organisational structure

The organisational structure reflects the hierarchy within the organisation, indicating the roles and responsibilities of the employees within the organisation. It also displays how different roles relate to one another as well as the structure of the departments within the whole organisation. Below is the organisational structure of Mootledi Logistics, as shown in Figure 4.1:



Figure 4.1: Mootledi Logistics organisational structure

4.2.2 IT structure

The IT structure of Mootledi Logistics displays the hierarchy and roles of those involved in Information Technology (IT) as well as Business. Both IT and Business reports to their various managers, who report directly to the Chief Information Officer (CIO) as depicted in Figure 4.2. Below is the IT structure of Mootledi Logistics:

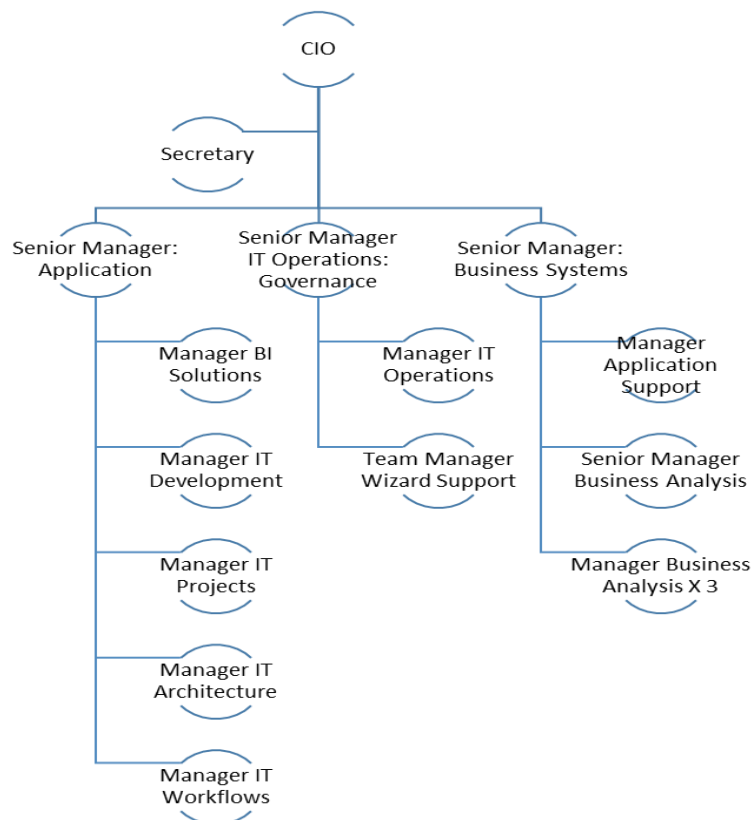


Figure 4.2: IT structure of Mootledi Logistics

4.2.3 Roles and responsibilities

Business Analyst

- Assisting with the business case
- Planning and monitoring
- Eliciting requirements
- Overseeing requirements of organisation
- Translating and simplifying requirements
- Overseeing requirements of management and communication
- Overseeing requirements for analysis

Software Developer

- Reviewing current systems
- Presenting ideas for system improvements, including cost proposals
- Working closely with analysts, designers and staff
- Producing detailed specifications and writing the program codes
- Testing the product in controlled, real situations before going live
- Preparing training manuals for users
- Maintaining the systems once they are up and running

Enterprise Architect

- Ensuring that technological goals of the enterprise are in line with the business goals
- Checking that the quality and reusability of enterprise software is convincing to ensure prospective cost savings with Service Oriented Architecture (SOA)
- Communicating comfortably the good and bad of a project and its deadlines
- Aligning efficiently the business goals with IT infrastructure supporting those goals
- Understanding the enterprise business while probing gritty IT issues

Business Intelligence Personnel

- Developing information architecture
- Managing current and future needs for data design and content
- Resolving semantic discrepancies in data definitions that arise among multiple sources and projects
- Creating reports
- Gaining consensus among users on common business data definitions

4.3 Overview: mmuso technologies

Mmuso Technologies was founded in April 1999 to combine and manage the government's information technology (IT) resources to save cost through scale increase delivery competences and improving interoperability. The Centurion branch has 450 employees, within which the IT department has 55 employees. This is a government institution responsible for providing information technology services to other government institutions. Its mandate is to improve service delivery to the public through the provision of information technology, information systems and related services in a maintained information system security environment to departments and public bodies. It also promotes the efficiency of departments and public bodies through the use of information technology.

Most government departments had internal IT facility divisions, which to a smaller or larger degree performed this work. The bigger departmental IT divisions tended to provide more services than the smaller divisions. A great deal more was contracted out to the private sector. Large numbers of departments were unable to recruit suitably qualified, experienced or knowledgeable staff to perform these functions and were forced to either contract the work out to the private sector or recruit consultants. In many cases, these consultants became full-time 'employees' at considerable cost to the department. The consequent over-dependence of government on contractors and alluring reduction of costs for services rendered are two key issues for resolution by Mmuso Technologies.

The primary reasons for the creation of Mmuso Technologies were the government's difficulty in recruiting, developing and retaining skilled IT personnel; managing IT procurement and ensuring that the government gets value for money; using IT to support transformation and service delivery; utilising expensive IT resources; and integrating IT initiatives. The organisation is situated in Gauteng, Centurion, with other branches throughout other provinces.

4.3.1 Organisational structure

Mmuso Technologies has its own unique organisational structure, depicted below in Figure 4.3:

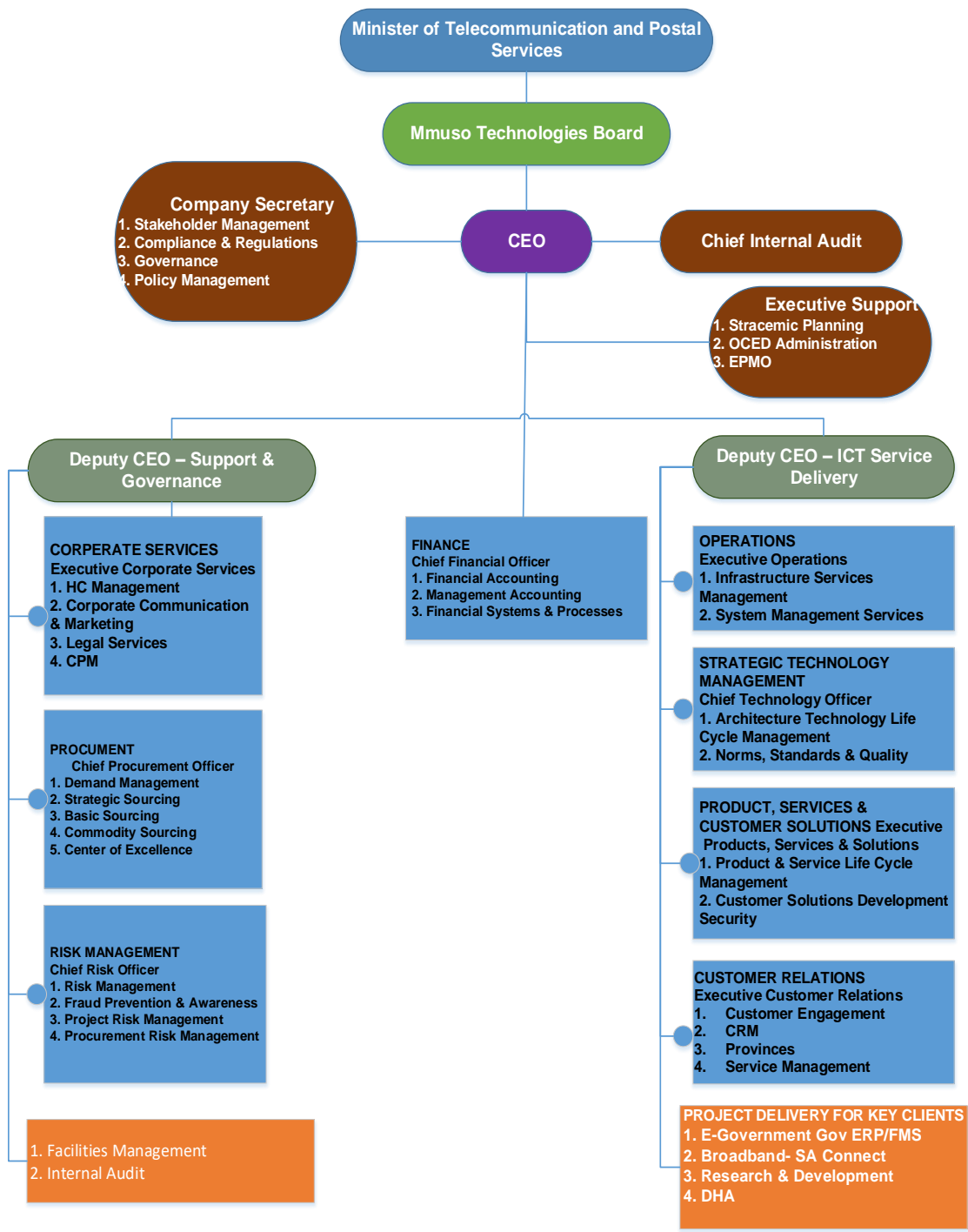


Figure 4.3: Mmuso Technologies organisational structure

4.3.2 IT structure

The IT structure of Mmuso Technologies reveals the hierarchy and roles of those involved in Information Technology (IT) as well as business. The focus is under the divisional Head of Norms, Standard & Quality which consists of various departments including Norms & Standards, Advisory Services, ICT Certification & Accreditation, Testing & QA and User Support Materials, as shown in Figure 4.4 below:

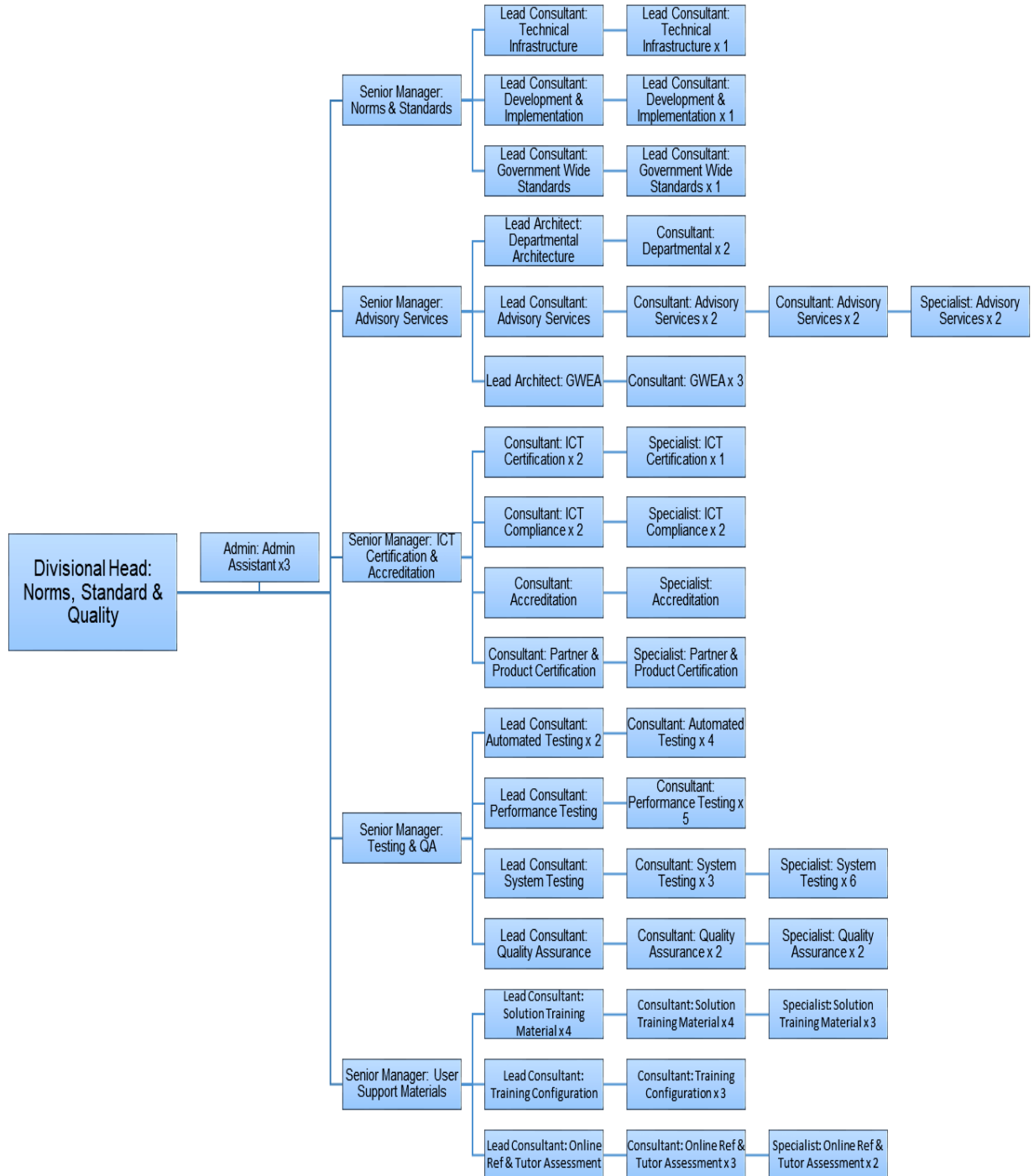


Figure 4.4: IT structure of Mmuso Technologies

4.3.3 Roles and responsibilities

Test Analyst

- Writing and executing test scripts for software and systems to detect faults
- Keeping written records of defects and bugs discovered during testing
- Analysing the defects and bugs to identify what is causing them
- Recommending solutions to fix any problems discovered during testing
- Tracking the success of the solutions
- Keeping software and systems documentation up to date
- Testing Lead/Senior: a Test Lead or Senior will often have the same responsibilities as a Test Analyst but will often have the added responsibility for mentoring junior members of the team and keeping the testing on track when the Test Manager is unavailable. They should definitely be writing their own test scripts.

Test Manager

- Showing team/ people management
- Defining the test strategy, then leading, guiding and monitoring the analysis, design, implementation and execution of the test cases, test procedures and test suits
- Deploying the appropriate testing framework to meet the testing mandate
- Defining the scope of testing within the context of each release/delivery
- Estimating the testing and negotiating to acquire the necessary resources
- Ensuring the test environment is set before and during test execution
- Communicating with others in the business to report on the test status (be it Project Managers or the senior leadership team).

4.4 Overview: Bokamoso Solutions

Bokamoso Solutions, situated in Illovo, Johannesburg, was established in 2010 and current has 30 employees. 18 of whom are in the IT department. The organisation invests in research to ensure its relevance in the industry, providing solutions that add value to its market base. The organisation offers specialised services in the software domain tailored to client needs, backed up by years of extensive practical experience which guarantees high levels of quality in everything they do. Bokamoso Solutions prefers to work *with* an organisation, not *for* an organisation, to deliver concepts, designs, installations, day-to-day operations and administration, capacity planning and longer-term growth strategies. Bokamoso Solutions offer the following services: custom software development, infrastructure management, software testing and consulting.

Custom Software Development

Bokamoso Solutions are the suppliers of business applications using in-house expertise, with extensive expertise in both the Java and MS.NET technologies, across all sizes of business applications, including mobile applications. The organisation's relational database management systems expertise ranges from MS SQL Server, Oracle, MySQL and DB2.

Infrastructure Management

They distribute equipment through partnerships with reputable hardware vendors, ranging from light-duty office equipment such as desktop PCs, notebook PCs, printers and facsimile machines, to heavy duty servers suitable for handling high transaction volumes. They not only distribute equipment but also provide setup and installation services.

Software Testing

Bokamoso Solutions provide software and systems testing services based on internationally acceptable standards such as IEEE 829 and IEC/EIA 12207. Their testing service includes (but is not limited to) the following elements: manual functional testing, automated regression testing, performance testing including load and stress testing, security testing, usability testing, backup and recovery testing, and data migration testing.

Consulting

They provide the following: analysis and advisory, project management, architecture and design, data management and data governance, and business intelligence.

How the organisation works

Bokamoso Solutions calls their work model an *Engagement Model* as they bring to other businesses the two most important pillars to those businesses. They bring to other businesses a strong enough software team with the necessary skills to build custom software. They design their own software solution according to specific business operations so other businesses can garner a competitive advantage relative to its competitors. Its custom fit solutions fit a custom budget, as they build according to special needs, supporting others with software building specialization. What this means is that customer do not pay for functionalities they do not need or use. Every cent paid is towards archiving special needs.

4.4.1 Organisational structure

The organisational structure for Bokamoso Solutions indicates the roles and responsibilities of the employees within the organisation. It also displays how different roles relate to each other as well as the structure of the departments within the whole organisation. Below is the organisational structure of Bokamoso Solutions, as shown in Figure 4.5:

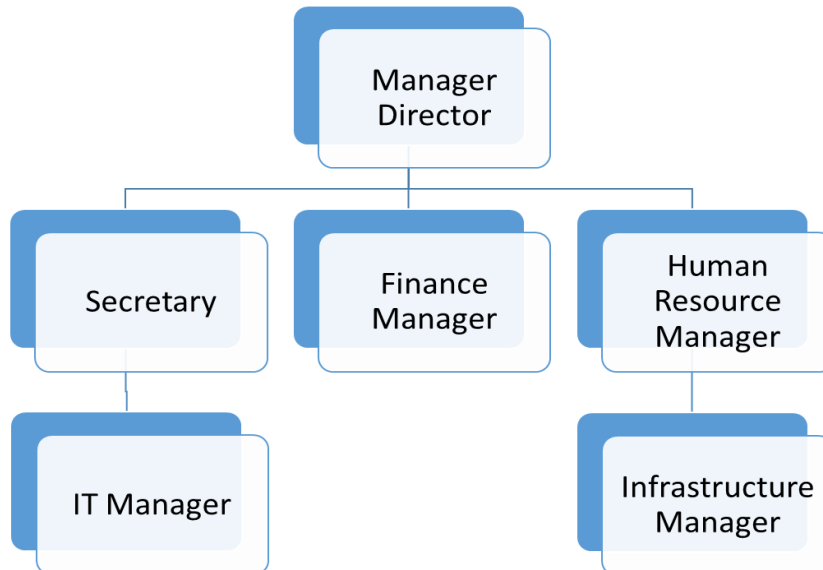


Figure 4.5: Bokamoso Solutions organisational structure

4.4.2 IT structure

Within the organisation, the IT unit has its own structure. The structure was divided into eight main units, as shown in Figure 4.6. The primary functions of each of the units are stated below:

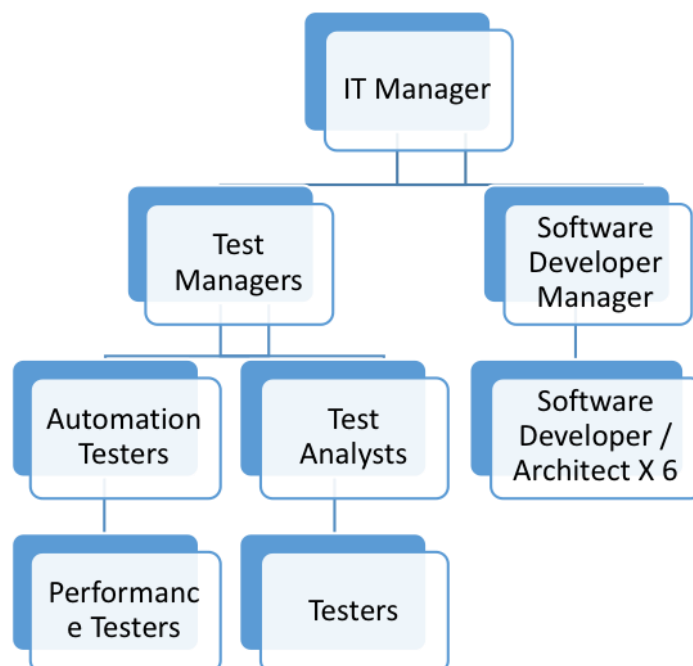


Figure 4.6: IT structure of Bokamoso Solutions

4.4.3 Roles and responsibilities

Automation Tester

- Work as part of cross functional, passionate agile project team to ensure quality is driven into the heart of the development process from requirement definition to delivery
- Designing, developing and supporting frameworks for test infrastructure and providing automation expertise to development teams
- Contributing toward predictable delivery of quality releases
- Research, recommend and implement tools as needed with the goal of increasing automation
- Mentoring team members on automation
- Proactively bringing problems to the attention of the team
- Generating and implementing innovative solutions to solve problems
- Being meticulous about documentation
- Maintaining a robust log of all test cases and test results
- Ensuring that all tests are executed and give regular feedback to the team lead on the status of quality
- Using appropriate measures and KPIs to validate software quality
- Working closely with the software engineering team, Product Management, Technical Operations, business users and Senior Management, as required
- Thinking creatively, quickly identifying and testing for functional 'edge cases' outside of expected functionality workflow.

Performance Tester

- Builds and maintains a scalable, portable, configurable automated testing framework designed using COTS and open-source tools as required
- Builds and maintains performance testing strategy and framework
- Front and back end performance testing, including WCF/API testing
- Performance, load, concurrent user and stress testing development, execution and publishing results
- Requirements and Functional Specifications review
- Test environment configuration/management
- Extensive troubleshooting in distributed/high availability environments
- Design and build intelligent test systems to simulate production load traffic
- Evaluate system performance, establishing baselines and identifying relative change on a build/release basis and potentially per customer basis
- Perform capacity, scenario, and endurance tests
- Analyse scalability, throughput and load testing metrics against test servers

- Compare and contrast system performance with varying levels of physical resources (RAM, CPU cores, Disk caches, Network) and compute nodes
- Execute performance optimization experiments
- Recommend short and long-term procedures

4.5 Summary

The three organisations used as case studies for this research – Mootledi Logistics, Mmuso Technologies and Bokamoso Solutions – are doing business in three different markets: transportation, information technology and power supply. The aim is to understand the way their information technology departments test and evaluate software. The various organisational structures as well as the information technology structures, including the roles of specific team members, have been highlighted. The relationships between various departments and roles have also been highlighted. In the next chapter, the analysis of data is carried out.

CHAPTER FIVE

DATA ANALYSIS

5.1 Introduction

This chapter presents the analysis of data from the three cases (organisations) under investigation in this study. As stated in Chapters 1 and 3, the objectives of the study were to examine and understand the methods applied in the testing of software. This includes examining the factors that influence software testing. Based on the results, a decision support system framework was created, which can be used to guide and improve testing and evaluation of software in organisations. This chapter is divided into three main sections and four sub sections. The main sections include case study, Actor Network Theory (ANT) and Diffusion of Innovation (DOI). The sub sections include actors, networks, moments of translation of ANT and the innovation decision process.

5.2 Overview of data analysis

As discussed in Chapter 3, three cases (organisations), Mootledi Logistics, Mmuso Technologies and Bokamoso Solutions were studied in this research. As discussed in Chapters 1 and 3, the analysis was carried out using two theories, Actor Network Theory (ANT) and Diffusion of Innovations (DOI). From the ANT perspective, the four moments of translation were applied for analysis. Complementarily, the innovation-decision process of DOI was applied. These two theories were employed to underpin the study primarily because of the vastness and extensiveness of the study. As explained in Chapters 2 and 3, it would have been difficult to achieve the objectives of the study, without gaps, if only one single theory was applied.

The focuses of these two theories are quite different. ANT focuses on actors (human and non-human), heterogeneity of networks and their relationships. Thus, ANT helps to understand how different networks are formed, as well as the interactions that happen between actors within heterogeneous networks. This makes it possible to gain an understanding of how actors got involved in the selection and use of tools and methods for testing. This includes gaining an understanding of the factors that influence the actions relating to software testing and evaluation. However, ANT does not explicitly focus on how technological artefacts such as software or the innovations (e.g. methods of use) implemented or diffused, which the DOI covers. Both theories, ANT and DOI have been presented at length in Chapter 2. The application of the theories was discussed in Chapter 3.

Another important factor was to determine which of the theories should be applied first in the analysis of the data. Iyamu (2013) discusses the criticality of order-of-use when theories are combined in a study, which assist to maintain logical flow, and for methodological value. In this study, ANT was applied first, followed by DOI, primarily because it was critical to first establish the formulation and existence of networks so as to know how the technology can be diffused in the environment. Also, it was necessary to first understand the tools, methods, and relationships between the actors involved in software testing, prior to assessment of innovative and diffusion.

For the purpose of data analysis, participants and organisations were labelled. As shown in Table 8 of Chapter 3, 14, 14, and 12 employees from Mootledi Logistics, Mmuso Technologies and Bokamoso Solutions, respectively, participated in the study. An example of the referencing standard is ML01, 5:1-3, which means organisation ML, participant 01, page number 5, and line numbers 1 to 3 for each of the cases are as follows:

- i. **Mootledi Logistics: participants - ML01 to ML14.** ML = Organisation name; 01 to 14 indicates the numbers of participants.
- ii. **Mmuso Technologies: participant - MT01 to MT14.** MT = Organisation name; 01 to 14 indicates the numbers of participants.
- iii. **Bokamoso Solutions: participants - BS01 to BS12.** BS = Organisation name; 01 to 12 indicates the number of participants.

5.3 Case study one: mootledi logistics

As discussed earlier, both ANT and DOI were applied as lenses in the data analysis. As explained in the overview section concerning order-of-use, ANT was applied first, followed by DOI.

5.4 Actor network theory

ANT treats actors equally, both human and non-human. Human actors are not treated superior to non-human actors. Various actors and networks were directly, indirectly, consciously or unconsciously involved in software testing activities within the organisation, Mootledi Logistics.

5.4.1 Actors

At Mootledi Logistics, both human and non-human actors were involved in the testing and evaluation of software. The human actors were from both the IT and business

departments. The actors had a common interest, to achieve the objectives of the organisation through software testing, evaluation and use. The actors from the IT department included project managers, business analysts, software developers and software application manager. From the business department perspective, the actors were business end-users, including business managers, and employees from finance, sales, operations, fleet services, human resources and risk management units. The non-human actors in the testing of software within the organisation fell into two categories: technical and non-technical, with technical actors including hardware, software and network. Some hardware for enabling and supporting software testing and evaluation included personal computers, servers and scanners.

Operating systems and software testing tools such as Mentis and Selenium were involved in the testing of software in the organisation. The network consists of local area network (LAN), wide area network (WAN) and WiFi. The non-human actors involved in the testing and evaluation of software in the organisation included documentation, process and methodology. The process that was followed was defined by the organisation. The methodology was agile method. One of the participants explained:

“We make use of N-Unit for integration testing, we use Selenium for automated web testing. Part of the agile methodology was followed in the delivering of software in an iterative and efficient manner” (ML01, 08:0283-0284).

Together, these actors had a common interest to produce software, using various tools and processes, which ANT also refers to as non-human actors. In ANT, human actors are not necessarily superior to non-human actors. For example, the software developers and software testers (human actors) rely on tools and processes (non-human actors) to fulfil their tasks. Without those tools and processes, such as Selenium and agile methodology, the software development and testing cannot be performed. Similarly, without the humans, the non-humans such as hardware, software and processes cannot by themselves produce software for the organisation. Therefore, collaboration between human actors and non-human actors made testing and evaluation of software possible.

5.4.2 Networks

At Mootledi Logistics, groups and units (networks) existed through which testing and the evaluation of software were carried out. These networks were divided into two main groups: the IT and business departments. Within the main networks, there were sub networks. Also, some networks replicated themselves within other, which ANT

refers to as *heterogeneity*. The networks were consciously and unconsciously formed. From the IT department perspective, the networks involved in the testing and evaluation of software were divided into two categories: technical and non-technical factors. The networks that focused on technical factors were the IT steering committee, a team of project stakeholders, a project management office (PMO), and a software development team.

The IT steering committee consisted of head of departments. The stakeholders were employees, which included product owners, project managers, IT managers and business managers. The PMO consisted of project managers within the organisation. Software developers, software project managers and software testers formed the software development team. Each of these teams or committees had roles, tasks assigned to them by the focal actors (software development manager) for reaching a common goal, to test and evaluate software in the organisation. However, some of the networks were in collaboration, as explained by one of the employees:

“The software development team and the business analysis team work together very closely and agree on certain parts of the software that needs to be released” (ML01, 10:0338-0340).

The other group of networks focusing on non-technical factors in the testing and evaluation of software included a team of business users consisting of employees from all departments and units. Similar to networks that existed within the IT department, these groups of business users had various roles, responsibilities and tasks they collectively undertook as assigned to them by the focal actors. Also, some employees were members of different networks which enact heterogeneity from an ANT perspective. For example, project sponsors and project owners were members of an IT steering committee and project management team and the team of stakeholders. The teams of software developers, business analysts and business users were responsible for software testing in the organisation. On the other hand, the IT steering committee was responsible for evaluating and approving changes or new projects logged by business:

“The requests coming to IT are recorded and there is now a change management process whereby if you want changes to be made you have to log them and they have to be approved by the IT steering committee” (ML02, 19:0673-0675).

These networks were formed to perform various software testing and evaluation tasks within the actor network. The networks were directly or indirectly connected through

individuals and groups, roles and responsibilities. This created heterogeneity of networks within the environment:

“Once the software development was complete the development team tested the software and when they were satisfied they handed it over to the business analysts to start testing. The user acceptance testing was conducted with the users to ensure that the software is acceptable to the users” (ML04, 25:0377-0378; ML01, 06:0202-0203).

However, there was no network or team that was specifically dedicated to software testing within Mootledi Logistics. As a result, other team members were occasionally assigned the responsibilities of software testing and evaluation. The individuals were from the business analysis team, software development team and business end users. Gradually, this group of individuals unconsciously formed a network. Actors and networks were inseparable during software testing and evaluation, in that a group of actors constitute a network, and networks were created by actors. In ANT, the actor can join more than one network at the same time, manifesting itself differently in each particular network (Iyamu & Tatnall, 2009). The networks were consciously and sub-consciously formed.

Also, there were networks which were heterogeneous in that they were in both IT and the business departments. For example, the team of senior management, which comprised of the CEO, CIO and departmental heads, were responsible for overseeing and making business decisions for both IT business departments. The team (network) formed part of other networks, which included project management, business analysis and software development teams.

5.4.3 Moments of translation

Through moments of translation, actors agree to the building of a network and acknowledge it to be worth defending. Translation serves as a mechanism of progressive temporary social orders or the transformation from one order to another through changes in the alignment of interests within the heterogeneous network (Gunawong & Gao, 2010). The translation occurs through four moments, which includes problematisation, interessement, Enrolment and mobilisation (Rhodes, 2009). These four moments of translation of ANT were used in the analysis of the data as a lens to zoom into the data that was collected at Mootledi Logistics.

Table 5.1: Moments of translation (Case 1)

<p>1: Problematisation</p> <p>The IT operations manager was the one who problematised the software testing and evaluation at Mootledi Logistics. The IT department needed new ways of how they can test and evaluate the developed software. During change management process meeting, it was raised that introducing the software testing techniques can help in developing and improving the software development team performance.</p>	<p>2: Interessement</p> <p>Not everyone was interested with what was proposed (software testing and evaluation). Some were worried of the change that software testing will bring to the software development team. The CIO understood what it meant to deliver quality projects, so he sponsored the project. Those who gained interest included business analysts, software developers and business users. Even the customers were also interested because they needed the software to serve themselves at any time they wanted to.</p>
<p>4: Mobilisation</p> <p>The software development and support manager encouraged software development team to research more about software testing tools so that they could be adopted to automate software testing. The use of these automation tools improved how software was tested within the organisation. Customers out there were also encouraged to use the tested organisation web site to serve themselves at their convenience. This shortened long queues that were experienced by customers at the rental counters.</p>	<p>3: Enrolment</p> <p>Business analysts participated by gathering business requirements and creating test cases so that the software could be developed and tested. Software developers converted the business requirements into functional software and wrote automation scripts to automate software testing using Selenium. Business users tested the software and signed it off. The project manager ensured that those who participated delivered their tasks on time.</p>

5.4.3.1 Moments of translation: problematisation

Mootledi Logistics relies on software for competitiveness and sustainability as software assists its employees to do their day-to-day activities as well as serving the organisations' customers. Therefore, any software that was developed within the organisation needed to be tested. Whenever a need arose to develop the new software, a certain process was followed within Mootledi Logistics: to log the request with the information technology (IT) department through the change management process. The change management process was a platform used to decide on whether the request made business sense. The project management team was responsible for evaluating the request and reported the outcome to the IT steering committee. Then, the IT steering committee's responsibility was to approve the requests based on whether it made business sense.

Those who were responsible for conducting software testing and evaluation within Mootledi Logistics included software developers, business analysts and business end users. It was vital for them to properly and rigorously test the software before it could be deployed to production for use. It is always better and less expensive to discover

faults while the software is still in the testing environment (because they could be fixed) rather than when it is in the production environment. Faults discovered by business end users in the production environment can have dire consequences on the image of the organisation. One participant stated that:

“With proper software testing we minimise faults coming back and forth which saves a lot of time and it enables us to deliver quality software” (ML09, 46:1665-1666).

Discussions regarding software testing and evaluation took place at the redline meetings that occurred within the organisation. The redline meetings improved the relationship between managers and their subordinates and between business and IT departments. It was an ongoing process of improving how things were done within the organisation, enabling all parties involved in the testing and evaluation of software to be on the same page (to have the same understanding). It also assisted managers in providing more information about the requests they have logged to avoid rolling out software that remained within the business without being used. The IT Operations Manager further stated that:

“I think the link between business and IT department is being defined much better than what it was before, and it would encourage the better output within the IT department” (ML02, 19:0699-0701).

Lack of processes within the organisation made it impossible to deliver the requested software. Previously the organisation had no proper quality assurance processes in place and the employees did not conduct thorough testing of the software developed. Poor analysis as well as no proper functional requirements specifications negatively affected the testing and evaluation of software. The software development and support manager highlighted that:

“We had no process previously and it was a question of let’s test this software a little bit” (ML01, 13:0454-0455).

The intention of delivering quality software was dependent on the IT department following proper quality assurance process. Failure in following due process resulted in deploying poorly functional software in production. In order for the IT department to deliver quality software, it was vital to adhere to the quality assurance process. One participant stated that:

“It had negative impact on the customers we serve which would ultimately cost the organisation money” (ML03, 23:0847-0848).

5.4.3.2 Moments of translation: interessement

Software testing and evaluation attracted the attention of various actors within the organisation whose interest was either of a voluntary or obligatory nature or both. The actors included individuals and groups from the management to operational level such as chief executive officer (CEO), chief information officer (CIO), IT steering committee, project management team, software development team, business analysis team as well as business end users. These actors had various roles and responsibilities in the testing and evaluation of software. Some were indirectly linked, and others directly linked to the software testing activities. The CIO, for example, was indirectly linked to software testing activities because he only sponsored the approved requests:

*“A sponsor being someone at the executive level who values the project and agrees to provide resources to execute the project”
(ML01, 07:0257-0258).*

The actors had diverse interests in the testing and evaluation of software within the organisation. Their interests were influenced by various factors. From the management perspective, they were expected to ensure that software was in place to enable business end users from various departments to perform their day-to-day duties, to serve customers and to provide customers with self-service software. Those departments included finance, risk and fleet department even to the point of Mootledi Logistics' customers out there.

*“The end users can be as broad as the public (customers), anyone who can go to our web site and reserve a vehicle from the web site”
(ML01, 03:0079-0080).*

The interest of teams and individuals was triggered by lack of quality assurance processes and standards especially for conducting software testing and evaluation within the organisation. Other teams, such as software development and business analysis, had their own standards which they followed when doing their tasks. When software had to be tested and evaluated, there was no software testing standard that was followed. The software developers, business analysts and business end users shared the responsibility of testing and evaluating software. Therefore, they followed their various team standards to conduct software testing:

*“The software developer's standard may not be exactly the same as the software tester standard or the business analyst standard because they may not know what exactly the expectations are”
(ML07, 45:1395-1396).*

Some software passed testing in the testing environment and failing in the production environment. Therefore, those responsible for software testing and evaluation needed to follow software testing processes and standards to test properly. Software testing is a career as much as project management, business analysis and software development, and it deserve the same respect shown to other careers:

“I think that software testing is not really given its due respect” (ML09, 43:1575-1576).

Some employees were interested in the software testing and evaluation because they were bound by the performance contract signed with their line managers on behalf of the organisation. These performance contracts were used to assess the performance of individual employees and teams concerning effectiveness and productivity, assisting managers to identify employee strengths and weaknesses. As a result, relevant training was provided to develop those who lack in skill. Performance contracts were also used to determine whether actors deserved salary increases or bonuses. Actors were expected to perform their duties as agreed with their managers:

“It is part of our annual employee performance” (ML01, 11:0381).

Other employees were interested due to the research that was conducted within the organisation regarding the investigation of software testing tools. These testing tools were open source and required no licensing, which were meant to make software testing and evaluation effective and efficient. Therefore, some software developers involved in this research gain more software testing knowledge and skills which could improve software testing and evaluation within the organisation:

“At the moment we are exploring the open source automation tool called Selenium. We plan to use it to automatically conduct our testing” (ML01, 01:0018-0019).

While several efforts were made by managers to stir interest for the proposed solution to the problematisation of software testing and evaluation, the building of interest among the employees could not be regarded a success. Not all actors who were interested participated in the testing and evaluation of software. Therefore, enrolment of employees was not completely successful as some employees were not willing to take part in the testing of software and evaluation.

5.4.3.3 Moments of translation: enrollment

Participation of actors was pursued through different means in the organisation. The managers of various teams engaged with their subordinates and negotiated their participation in the testing and evaluation of software. Redline meetings were held as

the negotiation platform to get those interested to enrol in the network that was going to test the requested software. Hence, software testing couldn't exist in isolation; various teams, including project management, business analysis, software developers as well as business department collaborated their skills to play a vital role. The managers of these teams used the power bestowed on them by the organisation to entice employees to participate and accept the roles and responsibilities allocated to them during the testing and evaluation of software:

“The scrum master should be someone who can liaise with business, product owner, software developers and business analysts” (ML14, 68:2523-2524).

Even though there was no dedicated software testing team within Mootledi Logistics, some kind of software testing was conducted. Software developers conducted unit testing and end-to-end testing, business analysts conducted functional testing and business end users conducted user acceptance testing. This was to ensure that when the software was handed over to the business department for use, it functioned as expected. The responsibility of the IT department was to enable the business department to execute its day-to-day functions of serving customers:

“Currently we don't have software testing teams, so testing is conducted by software developers, business analysts, business users and even the production support personnel” (ML13, 63:2302-2303).

As stated earlier in the previous stage, interest from employees was either of a voluntary or obligatory nature, or both. The managers of various teams used the power bestowed on them by the organisation to get employees to participate and accept the roles and responsibilities allocated to them during the development, testing and evaluation of software. These managers used their managerial discretion to enrol the employees in the network responsible for testing and evaluating software:

“The managers from different teams got to decide who they wanted to place on the project based on their skills” (ML10, 47:1719-1720).

The IT managers ensured that they enrol competent, skilled and dedicated employees who would fulfil their roles in the testing and evaluation of software. Mootledi Logistics relied on software for competitiveness and sustainability. Therefore, new software was developed, and existing software upgraded when a need arose. It was important for managers to have strong teams to fulfil these needs:

“There is diverse software that require testing within our organisation, therefore, we conduct different types of testing in order to test and evaluate them” (ML01, 01:0010-0011).

Without quality assurance processes in place, it was difficult for those who were testing and evaluating software to deliver quality software. Previously, the organisation was following the traditional methodologies, such as waterfall, to develop, test and evaluate software. They used to encounter challenges such as changes in requirements, business users not knowing exactly what they want, software taking too long to be completed and late inclusion of business end users in the software development life cycle. Therefore, re-examining user requirements and re-developing the software cost the organisation both time and money. Now the organisation has moved into agile software development such as scrum to improve on the challenges which they previously faced:

“Traditional methodologies are not as flexible as agile because with waterfall if a requirement is missed or has changed you have to wait until the software is deployed to production and then log a change request” (ML10, 50:1832-1835).

It is through functional software that Mootledi Logistics is able to function and serve its customers. Therefore, the IT department’s responsibility is first and foremost to deliver quality functional software. Mootledi Logistics seems not to be concerned much about performance testing. Performance testing is another important aspect of software testing and evaluation for determining how the software performs in terms of responsiveness and stability under a particular workload. There were no tools within the organisation that enabled the software developers to check the performance of the software on the network. Performance testing provides confidence that when the software is deployed to production it would be able to respond well under strenuous load on the network. However, software developers were improvising the performance element within the code they were writing when developing the software:

“I take much time to actually make sure that the efficiency in the processing is there so that the code I am writing will not slow the software.” (ML12, 58:2098-2100).

The most important thing within the organisation was to ensure that the software functioned as expected and business continued as usual. Enrolment was a success because actors accepted their roles and worked cooperatively towards delivering the tested and evaluated software, despite the challenges.

5.4.3.4 Moments of translation: mobilisation

Pursuance of employees was carried out along the structure and channels as defined by the organisation. In accordance with the organisational structure the product owner requested for the software, the IT steering committee decided on whether the software made business sense, and when approved, it was handed over to the IT department to develop, test and evaluate the software:

*“Business owner is someone who has requested for the software”
(ML14, 69:2524-2525).*

The CEO sponsored the entire initiative of developing, testing and evaluating the software. This software was going to be used by business users to serve customers and perform their day-to-day duties. The managers of various teams persuaded their subordinates to be part of the network to deliver quality software. The scrum master mobilised teams by ensuring that they delivered accordingly and also reported progress to the business owner:

“The scrum master and the team held sprint meetings to plan and communicate about the activities that needs to be done” (ML13, 64:2330-2331).

Persuasion was very important from all the above-mentioned parties because software development is a joint effort. As a result, everyone had to deliver on the tasks assigned to them on time in order to deliver the tested and evaluated software. Failure in doing so would negatively impact the delivery of the software. Some actors in the network were able to manage themselves and deliver accordingly whilst other had to be managed and checked on in terms of how far they were with their deliverables:

“As a project manager I track, monitor and check on the progress to ensure that the team is meeting the deadlines, they are on schedule and everything is moving as it should all the way from start until we hand the project over to business” (ML10, 46:1687-1689).

Mobilisation occurred through various platforms, including redline meetings as well as scrum meetings. Redline meetings used to be attended only by managers to discuss progress and issues within the project. Business users were formerly excluded but now they are included in those meetings because managers were unable to provide some crucial information regarding projects. Due to agile adoption, now there are scrum meetings attended by the members of the same team to discuss challenges and progress:

“During the scrum meeting everyone has to report about what they have done and what they are going to do” (ML14, 69:2518-2519).

It was always helpful to provide feedback to the product owner in order to know the progress about the software requested. Such feedback was also critical for project sponsors because no one wanted to waste the organisation’s time and money on something that was not achievable or something that was failing. Therefore, feedback persuaded all the actors to do their best to deliver the tested and evaluated software. At the end of the day, the requested software had to be signed off and accepted by business owners *and* business users:

“Then there was user acceptance testing whereby the business user runs through the testing in order to accept the software” (ML01, 06:0211-0212).

Therefore, product owners had to mobilise their teams to make use of the software that had been thoroughly tested and evaluated. The business team became the focal actor because the testing and evaluation of the software is initially instigated by them. The management was encouraged by the task of mobilisation which was linked to their performance appraisals. Mobilisation was successful, and employees were excited about the contribution in the testing and evaluation of software.

5.5 Diffusion of innovation

Diffusion of innovation (DOI) theory explains how new ideas or ideas that are perceived as new are spread within and between members of a social system (Sampaio, Varajão, Pires & de Moura Oliveira, 2012). It starts as an idea that is eventually converted into a product. That product cannot be left hanging without being diffused to the target market for utilisation. This diffusion was possible through the innovation decision process.

5.5.1 Innovation decision process

Innovation decision process was used as a lens to zoom into the data that was collected from Mootledi Logistics. Rogers explained that the innovation decision process is a process whereby “innovative” technologies are adopted and diffused following a five-stage model (Fuller, Hardin & Scott, 2007). Those stages include knowledge, persuasion, decision, implementation and confirmation (Sang & Tsai, 2009). This theory was covered extensively in Chapters 2 and 3. DOI was utilised in this present research to understand the decisions made in software testing and how software innovation was diffused within the organisation.

5.5.1.1 Innovation decision process: knowledge

The IT department had the enormous responsibility of delivering the software requested by the business department. The delivery of this initiative (software) was a new thing within the organisation; therefore, the IT department, which included project managers, business analysts, software developers and business intelligence personnel, needed to fully understand the business requirements to best deliver the tested, evaluated and functional software. The business analysts had the enormous responsibility of gathering the business requirements which would establish the development, testing and evaluation parameters for the requested software:

“The business analyst is the person who documented the business requirements the individual who have a clearer idea of how the software should function” (ML 10, 49:1786-1787).

Therefore, the business requirement specification, or functional requirement specification, provided absolute knowledge about what business required. However, this does not necessarily mean that the business requirements would be accurate at the first attempt. Sometimes business department would be uncertain as to what exactly they wanted. In such cases, the business analyst was able to assist as this person has both business and technical knowledge. There were also possibilities that some business requirements could be missed during the business requirement gathering process which would hinder the development and testing as well as the evaluation of software:

“If the software developer does not understand the business requirements they would automatically build incorrect software” (ML 14, 68:2490-2491).

The comprehension of the requested software was very important to the software development team as it enabled the team to develop, test and evaluate the software. It was important for everyone involved in the software development life cycle to have a clear and well-defined understanding of what business wanted. The software development life cycle process is a collaborated initiative that requires combined skills from various fields of specialisation such as business analysis, software development and software testing. For example, software testing cannot occur if the software does not exist; without business requirements software developers could not develop the software. Therefore, knowledge from business assisted business analysts to gather and document business requirements. As a result of the compiled skills and knowledge, the software development team was able to develop, test and evaluate the requested software:

“Business knowledge was also important to decide whether the test case was sufficient to actually test against the functional requirement” (ML01, 13:0445-0446).

Furthermore, the understanding of the existing software within the organisation helped the software development team to make informed decisions concerning the development of the new software as it was imperative that this new software integrate with the existing software. Undeniably, knowledge about this software was important:

“The knowledge of the systems that we have helps us a lot because we know what scenarios to test” (ML09, 45:1656-1657).

It would be virtually impossible to develop, test and evaluate software without understanding what was required. Therefore, it was crucial to understand the business requirements to accurately develop the software and identify scenarios to test. Those responsible for software testing and evaluation could not just test what they thought needed to be tested. They needed to have some sort of a guide for *what* to test and *how* to test it. Therefore, the business requirement specification assisted the software development team in that regard:

“One cannot test the software by just thinking that it should behave a certain manner, they needed to test it against the business requirements specification” (ML01, 13:0446-0448).

It was the responsibility of the software development team to seek clarity on vague business requirements in order to develop the correct software, as it is a challenge to develop and test software with ambiguous business requirements. Therefore, the software developers and software testers had to be innovative in clarifying vague business requirements in order to do precisely what was expected of them. There were instances, for example, whereby software developers were issued three pages of business requirement specifications which were not clear. As result, they had to liaise with business analysts as well as business users to find the necessary clarity. One of the software developers stated that:

“I would rather sit with the business users to get clarity, liaise with the business analysts and then start developing the software because jumping straight into software development without understanding what was required leads to back and forth which wastes time” (ML14, 69:2502-2504).

5.5.1.2 Innovation decision process: persuasion

The IT department was somewhat unfamiliar with the software that the business department had requested. Therefore, the IT department put together the software development team that was purposed with developing, testing and evaluating the requested software. This team included a project manager, business analyst, software developers and business intelligence personnel:

“Since our organisation has decided to go the agile route, the project manager is assigned the dedicated team of five individuals that includes a business analyst, two software developers, the business intelligent personnel and the K2 developer” (ML10, 47:1721-1723).

This team was interested in the innovation (requested software) and it actively looked for related information in order to develop, test and evaluate this software. Each individual had to actively play their role so that the team could succeed in delivering quality software. Therefore, the business analysts liaised with business users in order to find out what they really wanted in order to document the business requirements. The entire software development team (scrum) was dependent on the business requirement specification to perform their duties.

Failure in working as a team would negatively impact the output as well as delay the delivery timelines of the IT department. Consequently, the product owner as well as product sponsors would be dissatisfied with the IT department delivering the software later than the agreed timelines. The business department only expected what they have requested on the agreed timelines. They wanted software that would make their lives easier in terms of performing their duties and servicing customers:

“With proper software testing we minimize faults coming back and forth which save a lot of time, money and it enables the IT department to deliver quality software” (ML09, 46:1665-1666).

A lot was at stake for the organisation because the development and testing of the requested software was sponsored by the management. Also, the requested software was intended to simplify day-to-day performance of duties by business users and increase customer convenience. This software was envisioned to enable customers to help themselves any time of the day with the service they required from Mootledi Logistics. Therefore, it was imperative for the IT department to rigorously test and evaluate the software prior to its implementation in production:

“Roll backs in production costs time and money, it is better to test the software now while it is on the testing environment” (ML14, 70:2539-2540).

Finding defects in production puts the organisation in disrepute, especially when those defects are found by the customer. Defects found in production bring business to a standstill because the software ceases to be functional. Business end users are not able to perform their daily duties as well as rendering services to customers. Moreover, customers are prohibited from booking vehicles online. As a result, the organisation is depicted as unprofessional, as failing to provide services to its customers.

5.5.1.3 Innovation decision process: decision

Neither business end users nor customers could use the innovated software without having to test and evaluate it. Any software that is promoted to the production environment needed to be rigorously tested and evaluated to avoid mishaps in production. Hence, business users were required to conduct *user acceptance testing* (UAT) before the software could be promoted or deployed to production environment. UAT is the final stage of the software testing process, the process whereby business users test the software to ensure that it can handle required tasks in real-world scenarios, according to specifications:

“After the actual software testing the business analyst goes through the whole process with the business user to the point whereby the business user accepts the solution, which we call user acceptance testing” (ML05, 28:1026-1028).

However, UAT is preceded by *system integration testing* (SIT) which helps the software development team to verify whether the subsystems constituting the software solution work as expected and cooperate in a streamlined manner. A variety of software testing needed to be conducted for delivering software that met business requirements. During software development, software developers conducted unit testing which was termed *developer testing* at Mootledi Logistics. Unit testing is a software development process in which the smallest testable parts of the software, called units, are individually and independently tested for proper operation. This type of testing increased the confidence of software developers that when handing over the software to the business analysts and business users to test, it was fully functional. However, this was unfortunately not always the case at Mootledi Logistics:

“The biggest challenge is that the software developers don’t test the software properly the first time because they are expected to test it before deploying it to the testing environment but when the software is handed over to the business analysts to test it just keeps on failing” (ML07, 37:1354-1356).

The organisation had a definite need to improve how software was developed, tested and evaluated. Previously, the organisation followed traditional methodologies such as waterfall to develop software. Some participants stated that the waterfall methodology was too rigid, business requirement gathering and compilation of the requirements specification took too long to be completed, business users were involved too late in the software development process, and business requirements changed while the software was being developed. Also software developers, software testers, business intelligence personnel and support personnel had to wait for the requirements specification to be completed before performing their duties. As a result, by the time the software development was completed, the technology had already changed or advanced. Hence one software developer asserted that:

“The biggest issues in technology is that by the time you are done with the waterfall project the business rule has changed so what you are delivering doesn’t apply anymore” (ML12, 59:2168-2170).

Due to the above waterfall weaknesses, the organisation decided to adopt scrum agile methodology. Scrum is designed for teams of between three to nine software developers who break their work into actions that can be completed within sprints. The scrum team holds daily scrums (stand-up meetings) to report on progress, issues and the way forward. With agile methodology, business users were involved from the beginning of the project because they needed to verify all the functionalities they had requested. The changing of software development methodologies as well as the use of software testing tools were aimed at improving software development, software testing and its evaluation. This was the step in the right direction in terms of delivering quality software.

5.5.1.4 Innovation decision process: implementation

The software gets deployed to production once the business end users and product owner has accepted and signed it off. Thereafter, the software developers prepared the installation manual guide to assist the production support team in deploying the software. The production support team configured the production environment to accommodate the new software. Once the software was deployed, it was ready for use.

However, after deployment, the software needs to be tested to ensure that it functions as expected on the production environment before handing it over to business end users for use. The responsibility of the production support team was to ensure that the new software was always available for use and functional at all times. They also maintained the existing and new software, assisted business users with technical

production matters, carried out necessary research and provide in-depth analysis for resolving production issues. They were also expected to know and understand how all the software they maintained functioned. One business analyst stated that:

“At the end of the day the production support needs to support the software” (ML07, 39:1418).

The manager who requested had software for their department had to use the power bestowed on them to encourage staff to actually use the new software. Staff needed to understand, that the software was meant to simplify the day-to-day duties as well render services more efficiently to Mootledi Logistics customers. At Mootledi Logistics, any problems arising within the department needed to be attended to as soon as possible to ensure the efficient running of the department. Unresolved problems within the department will certainly have a negative impact on the organisation itself. Every department within the organisation has to function effectively for the entire organisation to be competitive.

5.5.1.5 Innovation decision process: confirmation

At this stage, it is critical to recognise and appreciate the benefits of using the innovation (new software), making the software part of the ongoing routine and promoting it to others within the department. This gives the department the opportunity to function more effectively. Business users are able to perform their daily duties using the new software as well as rendering the service to customers. One participant highlighted that:

“Some of the software at our rental counters are used by our staff to serve the customers” (ML01, 03:0077-0078).

Some of this software, such as the organisation web site, enables customers to serve themselves. Customers are able to make bookings and reserve cars through Mootledi Logistics web site, at their personal convenience, without having to be physically present at the Mootledi Logistics premises. The organisation had to develop efficient software that would enable it to maximise profits. Another participant asserted that:

“The organisation must ensure that the software it implements saves time and make the work environment friendlier” (ML06, 34:1247-1248).

Business user output indicated that the organisation made the right decision by implementing this new software. The organisation was heading in the right direction by introducing this new software which enhanced efficiency of services. The IT Operations Manager confessed that the organisation is not yet there but:

“I think we are heading in the right direction hopefully we would get there” (ML02, 19:0682-0683).

The individual finalises his decision to continue using the innovation. This stage is both intrapersonal (may cause cognitive dissonance) and interpersonal, confirmation the group has made the right decision.

5.6 Case study two: mmuso technologies

Similar to the first case of Mootledi Logistics, ANT was first applied to Mmuso Technologies, followed by DOI.

5.7 Actor network theory

5.7.1 Actors

The implementation of software within Mmuso Technologies was carried out by both human and non-human actors. The human actors were from IT and business departments, respectively. Through their combined efforts they were able to test and evaluate the requested software by other government departments. From the IT department there were project managers, business analysts, software developers, software testers and functional support personnel. Alternatively, business users and representatives from other government departments were from business, responsible for requesting the software that would enable them to perform their duties. However, business people worked closely with the business analysts to provide business requirements:

“We gather the business requirements for a change or a new project” (ML07, 17:0664).

These requirements were later converted into the requested software. The collaboration of actors' skills contributed immensely towards the development, testing and evaluation of software within the organisation. Clearly, software testing cannot occur in isolation. Other activities needed to occur prior to the existence of software such as gathering of business requirements, translation of those requirements into functional specifications and then the development of the requested software:

“We needed to work together as a team” (ML04, 10:0358-0359).

Those skills were dependent on each other to deliver quality software. However, human actors required non-human actors in order to perform software testing and evaluation of the requested software. The non-human actors were threefold: hardware, software and process. Hardware included personal computers, laptops, servers, external hard-drives and memory sticks. In order for hardware to function, it required software such as operating systems, programming languages, databases,

software testing tools and unified modelling languages. The organisation was still following the traditional methodologies or processes for testing and evaluating software:

“We are not yet using agile methodologies, we are still following traditional methods such as waterfall model and the V-Model” (ML05, 12:0468-0469).

The V-Model presented software testing at each stage of software development. Therefore, software testing did not have to be conducted at the final stages of the project. Each stage on the V-Model has a corresponding software testing activity which assisted in identify missing requirements and incorrect design early in the software development life cycle. As highlighted earlier, the organisation was planning to introduce agile methodologies to improve how software development was done.

“We are using the V-Model and we are trying to integrate it with the agile model” (ML04, 10:0358).

These tools were used for capturing test requirements, creating test cases, test case execution, logging defects and reporting software testing status. Software testers needed to undergo training to be able to utilise these tools accordingly. These tools were used to perform manual software testing. Due to cost, Mmuso Technologies employed free open source tools for performance testing. The selection of this tool was based on various factors. The organisation looked at the capability and proficiency of the software testing team for the tool adoption:

“We use J-Meter as a tool to conduct performance testing which is primarily used amongst others to simulate virtual users that were required in an event that simulate concurrent login” (MT14, 33:1268-1270).

Performance testing is another crucial part of testing and evaluating software as it measures the quality attributes of the software such as consistency and resource usage. The software testing tools enabled the software testers to perform software testing to the best of their abilities. At the completion of software testing, the software testing team would be in a position to confirm the quality of the software, intelligently advising on whether the software was ready or not ready for production:

“Our organisation produced quality software for another government department and business users were happy to use the software” (MT04, 12:0453-0454).

The quality software was expected to be functional at all times to enable business users to perform their day-to-day activities as well as to deliver services to the citizens of the Republic without any interruptions.

5.7.2 Networks

Networks, which refer to groups, units and departments, existed at Mmuso Technologies. The networks deployed software that was developed, tested and evaluated. As mentioned earlier, these networks were consciously and unconsciously formed. The formations of these networks were from business and Information Technology (IT) departments. IT department was responsible for converting business requirements into the requested software:

“We rolled out Integrated Financial Management System (IFMS) that integrates supply chain management, human resource management, financial management, payroll and business intelligence” (MT01, 01:0040-0041).

However, this software underwent software development process as well as rigorous software testing and evaluation. Mmuso Technologies was responsible for providing other government departments with software solutions that enabled them to function effectively. The IT department consisted of various teams directly linked to each other as they were expected to deliver quality software. Collaboration amongst the teams played a vital role in the development, testing and evaluation of software:

“We needed to work together as a team including project managers, systems analysts, business analysts, test analysts and software developers” (MT04, 10:0358-0360)

These teams formed heterogeneous networks within the organisation. The software project teams were derived from the above teams for every software project that was requested by another government department. The stakeholders within the software project team worked together to deliver quality software. The stakeholders from the management perspective included project owners and sponsors. From the operations perspective, the stakeholders included business analysts, software developers and software testers:

“So collectively the stakeholders included business which represent the government department that we were developing the solution for and others within the project includes the business analyst, systems analyst, software developers, the project manager and the other colleagues within our team we also regard them as project stakeholders” (MT14, 38:1471-1474).

The change management team was responsible for evaluating change requests logged, issues relating to those changes, the impact of those changes and evaluating new software projects to be developed by Mmuso Technologies.

5.7.3 Moments of translation

Table 5.2: Moments of translation (Case 2)

1: Problematisation	2: Interessement
<p>There were instances whereby Mmuso Technologies collaborated with vendors/consultants to develop some software requested by the government departments. The project manager problematised the formation of independent software testing team in order to verify the work done by the vendors. However, the challenge was that the software testers were unable to test and evaluate the software efficiently due to lack of knowledge and documentation of the software they were testing. As a result, software testing and evaluation appeared to be delaying the development of the requested software within the organisation</p>	<p>Some people within the organisation were excited by the formation of the independent software testing team. They thought that the software testing team was going to solve all the problems encountered during the software development. While others thought that software testing was just going to delay the deployment of software. Those who gained interest were from various teams including project management, business analysis, software analysis, software development, software testing as well as software support. The software testing team was going to ensure that government departments received quality software. As a result, those government departments supported the idea of having the independent software testing team</p>
4: Mobilisation	3: Enrolment
<p>Mobilisations of employees were carried out along this structure. Managers of various teams that were involved in the software testing and evaluation acted as spokespersons for their teams. The government department that requested the software became the focal actor because the testing and evaluation of software was initially initiated by them. Mobilisation was a success and employees were enthusiastic about what they were going to contribute in the testing and evaluation of software</p>	<p>Those who participated had a role to fulfil in the software development life cycle. They included project manager, business analyst, software developers, software testers and software support personnel. The release manager approved changes or new software that had made it through the software quality gate. The test manager provided necessary feedback the project stakeholders regarding software testing and evaluation. As a result, the release manager made use of the feedback received from the test manager to make appropriate decisions regarding the implementation of the changes or new software</p>

5.7.3.1 Moments of translation: problematisation

It is the responsibility of Mmuso Technologies to develop software for other government departments. Therefore, when any government department needed the software, they logged a request with Mmuso Technologies. This software was developed to enable the government departments to function efficiently and to render services to the citizens of South Africa. To accomplish this, software had to be rigorously tested and evaluated to ensure that it functioned as expected. There were

instances whereby Mmuso Technologies collaborated with external organisations and vendors to develop and configure other software, primarily due to lack of skills within the organisation. One participant alluded that:

“We wanted to have a total independent software testing team which will verify the work of the vendors” MT01, 03:0102-0103).

Due to the absence of independent software testing, the testing that was conducted was not intense. The software developers concentrated on doing software development and tested the piece of functionality they were assigned to verify whether it was working. Thereafter, it was assumed that the software was functional and ready for deployment, even without fully interrogating it. However, it is not easy for software developers to detect their own mistakes. So in many instances, the software testing team would only learn about the software when it was in production and already dysfunctional:

“In our organisation they are able to bypass software testing, as long as they want the software to be implemented it gets implemented without being tested” MT04, 11:0435-0436).

There was a change management as well as the release management team within the organisation. The change management team was responsible for evaluating the change requests logged, addressing issues relating to those changes and determining the impact of those changes on other projects. The release management team was responsible for approving changes and new software projects that needed to be implemented. Both teams, then, had processes and procedures that needed to be followed when dealing with changes or with new software projects needing implementation.

The other teams perceived the software testing team as a stumbling block or bottle neck regarding the deployment of changes and new software due to the release management process which required every single change or new software to be thoroughly tested and evaluated prior to deployment. Whenever the change or new software failed the software quality gates, it could not be approved by the release management. The strict release management process forced the software testing team to ensure that all changes and the new software was tested and evaluated before they could be implemented. One participant asserted that:

“It was important to make sure that processes were followed because in that case software testing would not be bypassed” (MT04, 11:0427- 0428).

According to the release management process, changes and software had to be rigorously tested in the Quality Assurance (QA) and Pre-Production (Pre-Prod) environments before they could be implemented, and they had to pass the necessary software quality gates. In one instance, to bypass this, one software tester spotted the business analyst changing the business requirement specification without consulting business:

“I think the challenge that we are facing in our organisation is that processes are not followed” (MT02, 06:0235- 0236).

As a result, the other teams worked against the software testing team. The release management process involved the software testing team to make crucial decisions regarding the deployment of changes and new software. Therefore, the other teams perceived software testing as problematic because of this process. The relationship between software testers and software developers was challenging, especially when defects were logged against software developers. One software tester highlighted that:

“When a software tester logs a defect against the software developer they would argue with you and even fight with you verbally” (MT03, 09:0317- 0318).

5.7.3.2 Moments of translation: interessement

The software development, testing and evaluation within the organisation attracted the attention of various actors. While some of these actors volunteered, others were obligated to partake in the testing and evaluation of software. From the management perspective, the actors included teams and individuals such as the change and release management teams, the business team, technology team, release manager, product owner, project managers, business analysts, software developers, software testers, and functional support personnel. These actors had various roles and responsibilities in the development, testing and evaluation of software:

“We need to work together as a team including project managers, systems analysts, business analysts, test analysts and software developers” (MT04, 10:0358-0360).

The actors had diverse interests in the testing and evaluation of software within the organisation, interests that were influenced by various factors. From the management perspective, they were expected to ensure that software produced enabled other government departments to perform their daily duties and render services to South African citizens. Some of those government departments included Environmental Affairs, the Development Bank of South Africa and Home Affairs. Each team from IT

and the business department would have a representative in the change management board:

“We would have a representative from the department who would represent the department that requested the software or change” (MT14, 38:1486-1487).

The government departments are not there to make profit but to serve the citizens of the country. Therefore, it was important for those departments to have quality software that would enable them to fulfil their duties as public servants. At the operational level, business analysts, software developers, software testers, functional support and business end users were liable to deliver quality software. Some of the factors that influenced their interest included enforcing processes and standards for conducting software testing and evaluation, involvement of all project stakeholders at the early stages of developing the software and promoting the culture of collective teamwork within the organisation:

“Actually what I believe needs to be done is to involve the software testers from the initiation of the project so that the tester can also give inputs and recommendations when coming to the testing of the software” (MT04, 11:0396-0398).

Some employees felt that it was human nature to bypass processes. Others felt that things could be done more quickly but the established processes prolonged things. So, even when processes were put in place, this did not necessarily mean that people would follow them. As a result, it was management responsibility to ensure that processes were followed. Failure in enforcing employees to follow processes would lead to Mmuso Technologies failing to produce quality software. Some participants stated that:

“Processes are not followed. It is important to make sure that processes are followed because in that case testing will not be bypassed. Processes are not followed at all and the management is doing nothing about the situation” (MT02, 06:0235-0584; MT04, 11:0427-0428; MT05, 14:0545).

Other employees wanted to be part of the software testing team that provided other government departments with quality software. One business analyst even left the business analysis team for software testing, desiring to learn to use software testing tools that were adopted within the organisation. Those tools included IBM Rational tools and JMeter, open source software used for performance testing. Mmuso

Technologies was lacking in test automations; therefore, they focused more on manual and performance testing:

“The reason why automation was not viable technology for us to improve on our testing capability the maturity of our testing team is not at a mature level where we are ready to automate” (MT14, 33:1262-1264).

As a result, individuals needed to be empowered in order to automate software testing. Not all actors who were interested enrolled, though, so the enrolment of individuals was not completely successful as some people were not willing to take part in the testing of software and evaluation.

5.7.3.3 Moments of translation: enrolment

Participation of actors was pursued through different means within Mmuso Technologies. The managers of various teams engaged with subordinates and negotiated their participation in the testing and evaluation of software. The team meetings were used as the negotiation platform to get those interested to enrol in the network that would test the requested software. Hence, software testing couldn't exist in isolation, various teams including project management, business analysts, software developers, software testing as well as business department collaboration of skills played a vital role. The managers of these teams used the power they had to encourage individuals to participate, to accept the roles and responsibilities allocated during the testing and evaluation of software:

“We have got a test manager who overlooks the testing capability who monitors the projects and plan and coordinate the testing activities and who work closely to the test lead” (MT14, 35:1345-1346).

The change and release management team existed within Mmuso Technologies. Any new software requirement or enhancement to existing software had to undergo the rigours of this team. Even after the software had been developed or enhancements made, these needed to be thoroughly tested and evaluated because this team relied on the test results from the software testing team. But even though this process existed, some employees still went ahead and bypassed processes and standards. The same participant highlighted that:

“You find out that the software was implemented without testing and we tested it while the software was in production those are the challenges we are facing” (MT04, 11:0414-0415).

The software testing team was faced with incessant challenges during the testing and evaluation of software. Some software developers did not have a good working relationship with software testers. Whenever software testers detected defects and logged them against respective software developer, arguments arose. As a result, software developers would think that software testers were attacking them or did not value the work they had produced. Meanwhile, project managers were committing to timelines without involving test managers. As the software testers were not always involved early in software development life cycle, teams involved in the software testing and evaluation were not treated the same. One software tester complained that:

“Another challenge is that all the projects that I worked on the development time could be extended but system testing time is never extended, and I don’t understand why” (MT02, 06:0202-0204).

The software testing team was treated unfairly because they had to sacrifice family time to work overtime to complete tasks assigned to them. Even so, they were perceived as enemies within the organisation. Even though they complained about how they were treated, management did not respond to their complaints. Mmuso Technologies adopted V-Model to develop software and enhance existing software. With the V-Model, each software development stage is countered by the software testing stage. After the delineation of requirements, the software testing team has to validate those requirements to create test requirements and test cases. Any ambiguous requirements needed to be clarified. The organisation also had software testing tools to capture all the testing activities.

5.7.3.4 Moments of translation: mobilisation

At this stage, employees acted on behalf of the organisation during the testing and evaluation of software in the organisation. Others were delegated to act as spokespersons to convince and persuade employees to participate in the delivery of quality software through its testing and evaluation. According to the organisational structure, the test manager was responsible for heading the testing team. Similarly, the change and release manager headed the change management board. The other members of various teams – such as project management, software development, and business analysts – reported to their respective managers. Mobilisations of employees were carried out along this structure. One participant explained as follows:

“We have got a test manager who overlooks the testing capability who monitors the projects and plan and coordinate the testing

activities and who work closely to the test lead” (MT14, 35:1345-1346).

The test manager appointed team leads who were responsible for individual projects. These team leads, assigned projects and three to four software testers depending on the scope of the project, were responsible for supervising the software testers assigned and also reported on the testing status of the project. Additionally, the team leads compiled their reports concerning their projects and submitted them to the test manager:

“Then they will do the execution and manage the day-to-day statuses and they will give the test manager a report on a daily basis” (MT14, 32:1245-1246).

The test manager would then gather all the reports from the team leads and consolidate them into one report which was then issued to the Head of Department. This process simplified reporting. The management was willing to accept the task of mobilisation which was linked to their performance appraisals. Mobilisation was a success and employees were enthusiastic about their contribution in the testing and evaluation of software.

5.8 Diffusion of innovation

Once the software was developed and tested it could not be left hanging without being diffused to the target market for proper utilisation. This diffusion was made possible through the innovation decision process.

5.8.1 Innovation decision process

Innovation decision process of DOI was used as a lens in the data analysis. This theory was covered extensively in Chapters 2 and 3. DOI was used to understand the decisions made in software testing and how software innovation was diffused within the organisation.

5.8.1.1 Innovation decision process: knowledge

Unlike business users who used one or two types of software, such as SAP or Oracle, to conduct their day-to-day activities, the software testing team had to test a variety of software. To this end, it was imperative for the software testing team to acquire knowledge about each and every software needing testing and evaluating. During the testing and evaluation of SAP, for example, there were no resources internally who understood SAP within the organisation other than vendors. Therefore, this lack of knowledge regarding SAP rendered it difficult for the software testing team to efficiently test and evaluate the software:

“I think software testers did not understand the functionality or know how to use SAP functioned which made software testing inefficient” (MT01, 03:0107-0109).

Vendors do not usually share knowledge about their work because this is how they generate money for their organisations or themselves. They intentionally keep certain knowledge to themselves so that they could be called in when something was not working. Even though that was the case, the test manager believed in upskilling his team members. The software tester’s responsibility was only to execute the test cases, log defects and re-test those defects. The test analyst, who was senior to the software tester, was responsible for extracting test requirements from the business requirement specification, creating test cases and capturing them on Rational Quality Manager. They could also perform the functions of the software tester. Therefore, at times the test manager would assign the software tester the functions of the test analyst:

“In my team I encourage cross skilling, where testers will fulfil the role of analysis so that they can grow towards becoming full fleshed test analyst but primarily we have got the test manager, test lead, test analyst and a tester” (MT14, 35:1351-1353).

Mmuso Technologies equipped the software testing team by sending them for training regarding the software testing tools the organisation had purchased. Not only that, the organisation also took its software testers on international software testing courses to learn the concepts, various types of software testing and how to apply those types of testing. This training greatly improved the skills of software testers and helped them perform testing efficiently:

“We attended International Software Testing Qualifications Board (ISTQB) which provides various testing courses, training on software testing and certification exams on those testing courses” (MT05, 14:0536-0537).

The software testing team acquired knowledge through testing various software within the organisation. As a result, their experiences were amassed through their skills.

5.8.1.2 Innovation decision process: persuasion

The IT department was persuaded by the change and release management to deliver quality software. All the various team members who were involved in the testing and evaluation of the software had to fulfil their roles to meet the timelines agreed upon with the particular government department requesting the software. It was not only

the responsibility of the software testing team to deliver on time, but the entire project team. Software testing and evaluation was a collaborated duty that relied on other skills for timely fulfilment:

“The requirements were gathered, software development took place, then software testing was performed then the software goes through the change management” (MT07, 18:0677-0678).

The business analysis team compiled the business requirement specification which was used to develop the software or make enhancements to the existing software. With these requirements, the software development team and software testing teams were able to perform their functions. As a result, this new innovation (software) persuaded the teams involved in the testing and evaluation of software to deliver quality software. However, there were challenges that occurred during the testing of the software:

“Not enough time was allocated to testing the project and we were pushed to complete our testing within a short space of time” (MT03, 08:0308-0309).

That, however, did not discourage the software testing team from doing their work. The software testers worked tirelessly in order to deliver quality software. They worked extra hours during the week and even came in on weekends to work. They never sat back and complained but worked as hard as possible because they understood that those who requested the software were expecting a finished product functioning as expected. One participant stated that:

“It was crucial that when you promise to deliver a product within the specified time let the delivery occur within the agreed time” (MT06, 16:0622-0624).

The change and release management was expecting the testing results from the testing team to make informed decisions about the tested and evaluated software.

5.8.1.3 Innovation decision process: decision

During software testing and evaluation, the software testing team decided to conduct various types of testing in order to deliver quality software. Once the team was satisfied with the quality of the software, they submitted the testing results to the change and release management. Based on the test results, the government department requesting the software was invited to conduct user acceptance testing. That was the final type of testing conducted with the customer (government department) with the help of the software testing team. The customer needed to test

the software to ensure that it functioned as expected. When satisfied, the department needed to sign off the software to confirm their satisfaction with the delivered software:

“The software testing team was involved in the user acceptance testing, we are the ones who facilitate it, we help the user with their acceptance criteria, the sign off” (MT08, 21:0806-0808).

The test results from the testing team were then issued to the change and release management team. The change and release management team used these test results to decide whether or not the software was ready for implementation. However, the change and release manager did not make all the decision alone; these were collaborative decisions with the managers of various teams within the IT department. One participant stated that:

“We formed a change management board and we signed off the change collectively from a business owner, software testing, and software development perspective so it was being cleared by the release manager who assumed a role of a chairperson in the change management board” (MT14, 38:1491-1494).

5.8.1.4 Innovation decision process: implementation

At this stage, the functional support team worked closely with the change and release management team as well as the software development team to implement the tested software at Mmuso Technologies. The functional support team needed the implementation plan created by the software development as well as the packaged software to deploy the software to production. The functional support team also had to configure the production environment prior to the deployment of software. The change management process still continued:

“If there was a change in the software, the change request was logged, and it followed all the necessary channels like it was developed, tested by software developers, tested by software testers and tested by functional specialist then it was deployed to production” (MT11, 26:0988-0990).

Once the software was deployed, the change management team altered the status of the implemented software to ‘complete’. The IT department at Mmuso Technologies officially handed over the software to the government department that requested the software. The product owner of the software notified their staff about the new software prepared for use. As expected, some employees were delighted to use the software and to even read more about the usefulness of the software, while others

were sceptical, too comfortable with how they did things prior to the new software deployment. One software tester asserted that:

“Our organisation produced quality software for development bank of South Africa and users are happy to use the system” (MT04, 12:0453-0454).

The manager who requested the software for their department had to engage the power bestowed with the position to encourage staff to use the new software. Staff needed to understand that the software intended to simplify their daily activities as well as to render efficient services to the citizens of the country. The staff also needed to understand that the organisation spent time and money in developing the software to ensure the government department was efficient in rendering these services to the citizens of South Africa. Therefore, the organisation had managers who were willing to engage their staff, especially when issues and conflicts arose.

5.8.1.5 Innovation decision process: confirmation

At this stage, it was necessary to recognise the benefits of using the innovation (new software), making the software part of the ongoing routine and promoting it to others within the department. As a result, this improves the government department’s opportunity to function effectively. The public servants who were end users were able to perform their daily duties with the new software as well as render the service to the citizens of the country. One participant highlighted that:

“Our organisation produced quality software for development bank of South Africa and users are happy to use the system” (MT04, 12:0453-0454).

The software is still being used to render services to the public. The software Mmuso Technologies developed enabled the government department to meet its mandate of delivering quality service to the citizens of the country. The end user output indicated that the government department made the right decision by implementing this new software. The government was heading in the right direction by introducing this new software, increasing efficiency. One participant praised the software testing:

“I am not saying the software testing team is the best of the best because even them they are in a growing phase, but I think it is a step in the right direction” (ML06, 15:0584-0585).

The government department finalised the decision to continue using the software (innovation), a stage of confirmation that the government department has made the right decision.

5.9 Case study three: bokamoso solutions

As described in Chapter 3, a brief description about Bokamoso Solutions was provided. Also in Chapter 4, a detailed overview of the organisation was presented.

5.10 Actor network theory

Various actors and networks were directly, indirectly, consciously or unconsciously involved in software testing activities within Bokamoso Solutions.

5.10.1 Actors

Bokamoso Solutions offers specialised services in the software domain tailored to the client needs. The organisation prefers to work *with* other organisations and not *for* those organisations. Bokamoso Solutions offer the following services: custom software development, infrastructure management, software testing and consulting. Bokamoso Solutions tests new software and enhancements made to existing software for their partners to ensure that all software is of acceptable quality prior to its deployment. There were combined efforts from human and non-human actors in the testing and evaluation of software. Both human and non-human actors have the ability to apply agency upon others, with neither being any more capable than the other (Lihosit, 2014). Bokamoso Solutions resources were based at the client site to efficiently provide the testing services they were employed to do.

The human actors were from both IT and the business departments, respectively. The actors had a common interest: to achieve the objectives of the organisation through software testing, evaluation and use. From the business department perspective, the actors were business end users, including business managers and employees from the respective department requesting the software. The actors from the IT department included project managers, business analysts, software developers, software testers and functional support. One of the participants stated that:

“The structure within the organisation is divided into different levels within the testing team, there are testers, test analysts, senior test analysts and test managers and they all play different roles within a project” (BS02, 05:0189-0191).

The non-human actors in the testing of software within the organisation were in two categories: technical and non-technical. The technical actors included hardware, software and network. Some of the hardware that was used to enable and support software testing and evaluation included personal computers, servers and mobile devices. Operating systems, software testing tools, such as HP Quality Center, Unified Functional Tester (UFT) and Load Runner were utilised in the testing of

software within the organisation. The network consists of local area network (LAN), wide area network (WAN) and WiFi. The testing and evaluation of software occurred on the mentioned networks.

The non-technical actors involved in the testing and evaluation of software in the organisation included documentation, process and methodology. The process that was followed was defined by the organisation. The organisation adopted the V-Model because it made it possible to involve a software testing team in the software development life cycle. Each stage of software development is linked to software testing in the V-Model. These actors had a common interest: to produce quality software through the use various tools and processes, which ANT also refers to as non-human actors.

In ANT, human actors are not necessarily superior to non-human actors. For example, the software developers and software testers (human actors) rely on tools and processes (non-human actors) to carry out their tasks. Without those tools and processes, such as Quality Center, Unified Functional Tester (UFT), Load Runner and V-Model methodology, the software development and testing could not be performed. Similarly, without humans, non-humans such as hardware, software and processes cannot by themselves produce software for the organisation. Therefore, collaboration between human actors and non-human actors made testing and evaluation of software possible at the organisation.

5.10.2 Networks

At the Bokamoso Solutions client, there were groups or networks that existed through which the testing and evaluation of software was performed. These networks were divided into two main groups: IT and business departments, respectively. Within the main network, there were networks formed. Also, some networks replicated themselves within other, which ANT refers to as heterogeneity. The networks were consciously and unconsciously formed, consciously in that the networks were officially created as part of the organisational structure. From the IT department perspective, the networks involved in the testing and evaluation of software were divided into two categories: technical and non-technical factors. The networks that focused on technical factors were comprised of a team of project stakeholders, a project management team, a software development team, a software testing team and a functional support team.

The stakeholders were employees, including product owners, project managers, IT managers and business managers. The project management team consisted of project managers within the organisation. Software developers, software testers and functional support personnel formed the software development team. Each of these teams had roles, specific tasks assigned to them by the focal actors (software testing functional manager) towards reaching a common goal, which was to test and evaluate software in the organisation. However, some of the networks worked in collaboration, as explained by one of the employees:

“We work in different teams, the software developers are a different team, the software testers are a different team, then people who compile specifications business analysts are a different team as well. So we are not one team therefore it is hard to coordinate all this teams” (BS01, 03:0112-0114).

Also, some employees were members of different networks, which enacts heterogeneity from an ANT perspective. For example, a software tester would part of the software testing team as well as the team of stakeholders. These teams were responsible for software testing and evaluation within the organisation. These networks were formed in order to perform various software testing and evaluation tasks within the actor network. The networks were directly or indirectly connected through individuals and groups, roles and responsibilities. This created heterogeneity of networks within the environment:

“When you are working together things go a bit smoother than when everyone is just doing their own thing” (BS01, 04:0137-0138).

Bokamoso Solutions was providing the full range of software testing services for their client. Actors and networks were inseparable during software testing and evaluation, in that a group of actors constitute a network, and networks were created by actors.

5.10.3 Moments of translation

As explained earlier in case 1 and case 2, the moments of translation were also used for case 3 as a lens to zoom into the data that was collected at the Bokamoso Solutions client.

Table 5.3: Moments of translation (Case 3)

<p>1: Problematisation</p> <p>The senior functional manager problematised software testing at Bokamoso Solutions. The manager was responsible for issuing projects as well as software tester to the test manager to test and evaluate the software within the organisation. Software was tested to ensure that business continued without any interruption.</p>	<p>2: Interessement</p> <p>Some employees were interested when notified about software that needed to be tested. Business analysts, software developers and software testers were interested and couldn't wait start testing and evaluating software that needed to be tested. They wanted to deliver quality software that was going to assist enable business users to perform their duties as well as rendering services to customers.</p>
<p>4: Mobilisation</p> <p>The product owners became a spokesperson because they had to come up with innovative ideas concerning new software. Managers of various teams such as business analysis, software development and software testing represented their teams in accordance to the organisational structure.</p>	<p>3: Enrolment</p> <p>Test managers participated by assigning tasks to software testers and reporting about testing status to the senior manager. Project managers worked with all project stakeholders in delivering the requested software by business. The software testing utilised software testing tools such as Quality Center to capture test cases. Automation testers used unified functional tester to automate manual test cases. Performance testers used load runner to test the performance of the software on the network.</p>

5.10.3.1 Moments of translation: problematisation

The Bokamoso Solutions client used software to make business, selling products and rendering services to its customers through software. Furthermore, other existing software was enhanced to keep business going as technology continued to advance. Therefore, the Bokamoso Solutions client had a list of software that needed to be tested. The senior functional manager was responsible for assigning those projects to test managers. One of test manager alluded that:

“The senior functional manager manages the whole software testing division, they oversee all the projects that are coming in for testing, they source projects which are to be tested by their division” (BS02, 06:0213-0214).

From time to time, a need arose from business to request the IT department to develop new software or enhance existing software for them. The process followed was for business to log the request with the information technology (IT) department in which a committee was responsible for approving new projects or enhancements to existing software. The demand management committee (DMC) committee validated the requests and the costs that would be incurred to develop, test and evaluate that

request (software). The change management process was a platform for deciding whether or not the request made astute business sense.

The DMC committee responsibility was to approve the requests based on whether it made business sense and the cost to be incurred. After approval, the request is handed over to the IT department to develop the requested software. Once the software was developed, it was tested intensely prior to deployment to production for use by business users. However, there were instances whereby the software testing team was not provided with the business requirement specification but expected to perform software testing anyway. One software test analyst suggested that:

“A rule needed to be introduced emphasizing that whoever was logging a request needed to provide documentation, if they didn’t have a documentation the testing team was not going to test their software, we are not going to assist you” (BS06, 29:1129-1131).

Without necessary documentation, it was difficult for the software testing team to plan software testing activities. The business or functional requirement specifications enable the team to extract requirements, identify scenarios to test and create test cases. Performing software testing without the stated documents was disastrous because the software testing team wouldn’t know what to test or what to cover. The developed software had to be tested against the business or functional requirement specification. The same participant stated that:

“It is many projects that you are not provided with documentation. Documentation is the biggest challenge within our organisation” (BS06, 29:1123).

Moreover, the business department had tendencies of pressurising the project manager to deliver the software on unrealistic timelines. As a result, such pressure negatively impacted the quality of the software because the project manager, software developers and software testers were pushed to complete their work in an unrealistically short period of time. At times, project managers estimated testing timelines without consulting the testing team, generating conflicts between project stakeholders and deteriorating the relationships among project stakeholders. One participant complained that:

“Actually the resources within the testing team at some point they are seen as inferior by other project stakeholders” (BS03, 18:0703-0704).

The best technology alone, good processes alone as well as skilled people alone cannot deliver quality software. Hence, there were networks formed by both human

and non-human actors collaboratively. Together, the network was capable of delivering quality software. The IT department just needed to follow the processes and utilise the available technology to produce quality software. As a result, business would be satisfied with what would be delivered to them.

5.10.3.2 Moments of translation: interessement

Various actors were enticed by the idea of testing and evaluation software within the organisation. Some actors' interest was voluntary while for others, interest was obligatory in nature, or both. The actors included individuals and groups from management to operational levels such as IDR committee, senior functional manager, project management team, software development team, business analysis team, software testing team, functional support team as well as business end users. These actors had various roles and responsibilities in the testing and evaluation of software. Some were indirectly linked while others directly linked to software testing activities:

“The role of the test manager would be to create the statement of work, test plan, oversee at what the test analyst and testers are doing” (BS01, 04:0142-0143).

The actors held diverse interests in the testing and evaluation of software within the organisation influenced by various factors. From the management perspective, IT was expected to deliver quality software that would enable business to continue without interruption. The requested software was needed by departments like human resources, finance, marketing and public relations to perform their work functions properly and render services seamlessly to their customers. Hence, quality was emphasised on all software that was tested and evaluated within the organisation.

As a result, those who were interested wanted to improve how things were done within SDLC. If the software was not developed properly it not only impacted the software development team but affected the entire team responsible for delivering quality software. When the business analysis team produced unclear requirements, this befitted the entire project team. Therefore, it was important for all the project stakeholders to work collaboratively in the delivery of quality software:

“It is much better to get people together to understand what each other is doing because they are in one team” (BS05, 27:1062-1063).

Other employees were interested because they wanted to learn how software testing was performed. They expressed a desire to learn to use software testing tools that were adopted within the organisation. Those tools included Load Runner, UFT, Quality Center in order to perform various types of testing. As a result, software

testing teams were strengthened and produced quality software every time they had to test and evaluate the software:

“People learn best at work when they practically perform activities assigned to them” (BS02, 08:0312).

Furthermore, individuals were empowered and in turn empowered other teams to be efficient and productive. Not all actors who were interested enrolled, though. Therefore, enrolment of individuals was not completely successful as some people were not willing to take part in the testing of software and evaluation.

5.10.3.3 Moments of translation: enrolment

Participation of actors was pursued through different means within the organisation. The managers of various teams engaged with their subordinates and negotiated their participation in the testing and evaluation of software. The team meetings were used as the negotiation platform to entice those interested to enrol in the network for testing the requested software. Hence, software testing couldn't exist in isolation: various teams, including project management, business analysis, software developers, software testing as well as the business department, played a vital role in the collaboration of useful skills. The managers of these teams relied on the power bestowed upon them by the organisation to convince individuals to participate and accept the roles and responsibilities allocated to them during the testing and evaluation of software:

“We have got test managers who oversee the overall progress of the testing team” (BS08, 37:1440-1441).

The software does not just exist but must be developed and tested before it can be delivered to business. This then required interdependency of skills such as project management, business analysis, software development, software testing and software deployment. Various employees within the organisation were required to fulfil those roles for successful delivery of quality software to business. There were project managers, business analysts, software developers and software testers:

“Performance testers overseas the non-functional requirements of the software like load, stress and performance of the system” (BS07, 33:1299-1300).

As stated earlier, interest from employees was either of a voluntary or obligatory nature, or occasionally both. The managers of various teams used the power bestowed on them by the organisation to lure employees to participate and accept the roles and responsibilities allocated to them during the development, testing and

evaluation of software. These managers used their managerial discretion to enrol the employees in the network responsible for testing and evaluating software:

“The test manager oversees the testing for projects assigned to them and reports to the senior functional manager” (BS02, 47:0208-0209).

The managers also enforced the tasks allocated to individuals through to organisation’s compulsory performance evaluation, and as such, other employees were forced to participate. The managers of various teams were under pressure to establish a high-performance team responsible to deliver the solutions for the business request. All that business expected from the IT department was to develop, test, evaluate and deploy the software that could enable them to perform their day-to-day duties for providing the best uninterrupted services to their customers. As a result, these managers had to ensure that they enrol competent, skilled and dedicated employees who would fulfil their roles in the testing and evaluation of software. The Bokamoso Solutions client relied on software for competitiveness and sustainability. Therefore, new software was developed, and existing ones upgraded when a need arose. It was important for managers to compile strongly skilled teams to fulfil these needs:

“People should have the right skills to avoid testing incorrectly because the software they are testing is going to be used business end users and not the tester” (BS08, 37:1428-1429).

These tools assisted the software testing team in terms of delivering the tasks on time and potentially even quicker. At least Bokamoso Solution performed testing for both functional and non-functional requirements. The functionality of the software can work properly but its performance, if not tested, can fail in production. The software should be able to perform no matter how many users are logging in simultaneously to the software. Thus, performance testing was conducted. Enrolment was a success because actors accepted their roles and worked together towards delivering the tested and evaluated software even though there were challenges.

5.10.3.4 Moments of translation: mobilisation

In the testing and evaluation of software in the organisation, employees acted on behalf of individuals, the team or the organisation in entirety. Others acted as a spokesperson to convince and persuade employees to participate in the delivery of quality software. In accordance with the organisational structure, the senior functional manager heading the software testing division in the organisation was responsible for the testing all software. The managers of various teams within the IT department reported to their respective managers. Mobilisation of employees was carried out

along the structure and channel as defined by the organisation. One participant alluded that:

“The structure within the organisation is divided into different levels within the testing team, there are testers, test analysts, senior test analysts and test managers and they all play different roles within a project” (BS02, 05:0189-0191).

The software testing team was responsible for ensuring that the software was rigorously tested and evaluated before it could be implemented to production. Therefore, the team conducted various types of testing and utilised the software testing tools existent within the organisation. Various team members had to play their part in the testing and evaluation of the software:

“Therefore, you have a project manager, product owner, software developers and software testers so everybody performed what they were assigned to do” (BS03, 12:0460-0461).

The product owners became spokespeople because they had to generate innovative ideas concerning new software or clever enhancements on existing software to enable the business to remain sustainable and competitive. The business end users had to sell products and render services to customers at all times without interruption. Therefore, the product owner worked very closely with the project team to ensure that quality software was produced for keeping the business going:

“There is a product owner who actually updates the team on what actually needs to be done” (BS10, 41:1617-1618).

Once the mobilisation was successful, then the actor network began to function with the objective of delivering the requested software by business. The business team became the focal actor because the testing and evaluation of software was initially instigated by them.

5.11 Diffusion of innovation

The diffusion was made possible through the innovation decision process.

5.11.1 Innovation decision process

The innovation decision process of DOI was used as a lens to zoom into the data collected from Bokamoso Solutions. This theory was presented extensively in Chapters 2 and 3.

5.11.1.1 Innovation decision process: knowledge

The delivery of quality software to business was not the sole responsibility of the software testing team but rather the entire IT department. It was a team effort to ensure that business received sufficient quality software. Therefore, a positive relationship among the project stakeholders was vital in terms of delivering good quality software. Software testing is a complex endeavour because software testers test not only single but multiple types of software. Therefore, software testers need to be fully equipped with knowledge about the software under test. Software testing is not all about detecting defects but also uncovering situations that could negatively impact the customer in terms of usability and maintainability. One participant asserted that:

“The software tester needs to know how much testing to perform that is critical, accuracy and understanding what they are testing and at least they know they are focusing on the right direction and producing the right results as expected” (BS02, 07:0236-0238).

Therefore, if the software tester does not have the necessary knowledge of the software, the testing will not be done sufficiently. They could easily miss some important functionality of the software. As a result, poor quality software would be delivered to business and the business would surely not be impressed with such software. Consequently, the entire IT department would be perceived as unprofessional, as a team that does not know what they are doing. For example, for both manual and performance testing, it is vital for the software tester and performance tester to understand what they need to test. With a failure in understanding the requirements, software testing would waste time, money and resources:

“So with performance testing we start by engaging with the user to understand the requirements for the software they want us to test” (BS04, 20:0774-0775).

Thus, business requirement specifications, functional requirement specifications, are clearly needed before software testing could commence. This document provided software testers with the necessary information and understanding of what business has requested. With such information, the software testing team was able to understand the scope of testing and plan accordingly. As a result, software testers identified various scenarios which were used to create test cases using tools adopted within the organisation:

“It was important to understand the business requirements, how the users were using the software and their expectations” (BS05, 26:0993-0994).

Software testers need to be innovative, analytical and good in communicating with the rest of the project stakeholders to elicit the relevant information to assist them in testing and evaluating software. Also, some departments at the Bokamoso Solutions client did not know exactly what the software testing team was doing. Therefore, during the monthly meeting, called *Imbizo*, the software testing team needed to prepare presentations about projects they had tested in the past and the ones they were currently testing to educate other departments concerning their role:

“So in those sessions I think that was where we pulled out items that we educated other team members that this was what testing was all about and the value it added to the organisation” (BS02, 09:0327-0328).

5.11.1.2 Innovation decision process: persuasion

The software testing team was persuaded to get clarity on ambiguous requirements which were in the business requirements. Therefore, other project stakeholders such as business analysts, software developers and business users were there to assist the software testing team with any unclear requirements to enable the software testing team to cover as many scenarios as possible in the delivery of quality software to business. The risk with ambiguous requirements was that business rules could be incorrectly built within the code and some important functionality could be overlooked, thereby being detected as a defect. Therefore, it was vital to detect such defects early in the software development life cycle:

“In most cases you find that the business analyst was not aware that they missed some requirements” (BS02, 10:0376-0377).

Therefore, through the assistance of the software testers, any missing requirements could be detected early. As mentioned before, human beings are prone to committing mistakes. Project stakeholders needed to work together to deliver quality software, as blaming each other will not rectify mistakes committed by other stakeholders. It was a challenge for various teams to interact smoothly with each other because different teams have their own ways of doing things. Therefore, it was imperative for the stakeholders to build a good relationship in order to deliver quality software on time:

“I kind like created a work relationship so it was easier for me to communicate with the project manager and also the other people that I needed information from” (BS01, 05:0173-0174).

Without good relationships among project stakeholders, it would have been difficult to test and evaluate software. Some stakeholders had information about things that others did not know about. There were specialists who had intense knowledge about their work. Therefore, those specialists were a good point of reference in clarifying requirements which were not clear to software testers:

“There are specialist who have been supporting or working on a software for a number of years, so they are key people who know the ins and the outs of the software” (BS01, 05:0173-0174).

As a result, consultations with those who understood the environment better assisted the software testing team in getting the information required for testing and evaluating the software.

5.11.1.3 Innovation decision process: decision

The software testing team was faced with numerous decisions regarding the testing and evaluation of software. For example, the team needed to decide which testing methods to adopt, how many cycles of software testing they needed to conduct and when and how to report on the testing progress. They also needed to decide which types of testing they were going to conduct, for example, manual, automation or performance testing. The intention was to cover the entire software testing scope for the software under test:

“So for us to decide whether we were going to automate software or not we looked at how stable enough the software was to be automated” (BS03, 17:0348-0349).

The software testing team had to further ensure that they performed performance testing on the software under test. Therefore, it was vital for the software testing to cover non-functional testing of the software. Software could work perfectly well functionally but fail on non-functional requirements. As a result, Bokamoso Solutions decided to conduct performance testing for all the software they tested to ensure that they also covered non-functional requirements. It vital to cover the response time, speed and stability of the software under test:

“Performance testers overseas the non-functional requirements of the software like load, stress and performance of the software” (BS07, 33:1299-1300).

It was crucial for software testing to cover both functional and non-functional requirements of the software under test. At the completion of software testing and evaluation, end users from business were invited to conduct user acceptance testing.

This testing was coordinated by the software testing team. The team also assisted end users during the testing whenever they needed help. This testing determined whether end users or the business were satisfied with the software:

“User acceptance testing, it is whereby the user was accepting that what was tested was what they have requested, and they were happy with the software” (BS01, 01:0029-0031).

5.11.1.4 Innovation decision process: implementation

Once the business end users were satisfied with the quality of software, they signed off the user acceptance testing. All the software testing documents signed off by various stakeholders involved in the software testing were stored confidentially as these documents were needed during the auditing period. The user acceptance testing closure report was one such requirement. When business end users were satisfied with the software, the next step was to deploy that software to production. As a result, the production environment was configured accordingly, and the software deployed:

“The production environment is a live environment where the tested application is deployed” (BS03, 14:0522).

The functional support team worked closely with the software development team and software testing team to deploy the software. The functional support team configured the production environment according to the implementation plan. Software developers packaged the software and also prepared the implementation plan. The software testing team conducted smoke testing after the deployment testing just to ensure that the software was still functioning as expected.

The software was deployed because business was dependent on it to sell products and render necessary services to customers. The software also enabled them to perform their daily duties. Many organisations rely on software to do business and remain competitive. Therefore, the IT department was the solution to the business request. The product owner of the software notified their staff about the new software that would be available for use. As expected, some employees were delighted to use the software and even to explore more about the usefulness of the software, while others were not. One software tester asserted that:

“There is a product owner who actually updates the team on what actually needs to be done” (BS10, 41:1617-1618).

Once the software was implemented, it was ready for use. The manager who requested the software for his department had to wield the power bestowed on him to

encourage staff to use the new software as staff needed to accept that the software was meant to simplify their daily activities as well as render efficient services to the customers. The staff also needed to understand that the organisation expended valuable time and money in developing the software to ensure the organisation was efficient in selling products and rendering services to customers. The Bokamoso Solutions client was revenue driven; therefore, the organisation had managers who were willing to engage their staff, especially when there were issues.

5.11.1.5 Innovation decision process: confirmation

At this stage, it is important to acknowledge the benefits of using the innovation (the new software), integrating the software into the company's ongoing routine and promoting it to others within the department. This provided the department with the opportunity to function more effectively. Business end users were able to perform their daily duties using the new software as well as render the service to customers. One participant highlighted that:

“We wanted to make sure that the end product was usable and was user friendly as our customers was our users of the software” (BS07, 31:1220-1221).

The business was able to revert to IT and confirm the outcome of the deployed software. Even when things were not working properly in production, business could still come back and report problems to the IT department. The IT department was available to ensure that business continued at all times. At the confirmation stage, the business finalised its decision regarding the software that was implemented, confirming how well the software was being used by business end users. In chapter six, the next chapter, the findings from the data analysis are presented. Also, covered in the chapter is the discussion of the interpretations.

CHAPTER SIX

FINDINGS AND INTEPRETATION

6.1 Introduction

This chapter presents the findings of three case studies: namely Mootledi Logistics, Mmuso Technologies and Bokamoso Solutions. Moreover, it presents the interpretation of the study. The objective of the study was to create the decision support system framework for testing and evaluation software in organisations. Based on the analysis and findings from the three case studies, interpretation was carried out from which a framework (Figure 6.4) was developed. The framework is aimed at gaining a better understanding and thereby addressing the challenges which occur during software testing and evaluation within organisations. The analysis was done using the moments of translation, the lens of actor network theory (ANT). Additionally, the innovation decision process of diffusion of innovation (DOI) was also used to analyse the data.

The remainder of this chapter is structured into six main sections. The first section covers the introduction. The second, third and fourth sections incorporate the findings and the discussions surrounding the three specific case studies. The fifth section presents the interpretation of the findings and discusses the framework for testing and evaluating software within organisations. The final section summarises the chapter.

6.2 Findings and discussions: mootledi logistics

Based on the analysis of the empirical data from case 1, Mootledi Logistics, five factors were found to have a critical influence on the software testing and evaluation within the organisation: (1) lack of testing framework; (2) lack of management buy-in; (3) network of employees; (4) quality of software; and (5) lack of standards and procedures. Figure 6.1 depicts these factors and their relation to each other. These factors are discussed below:

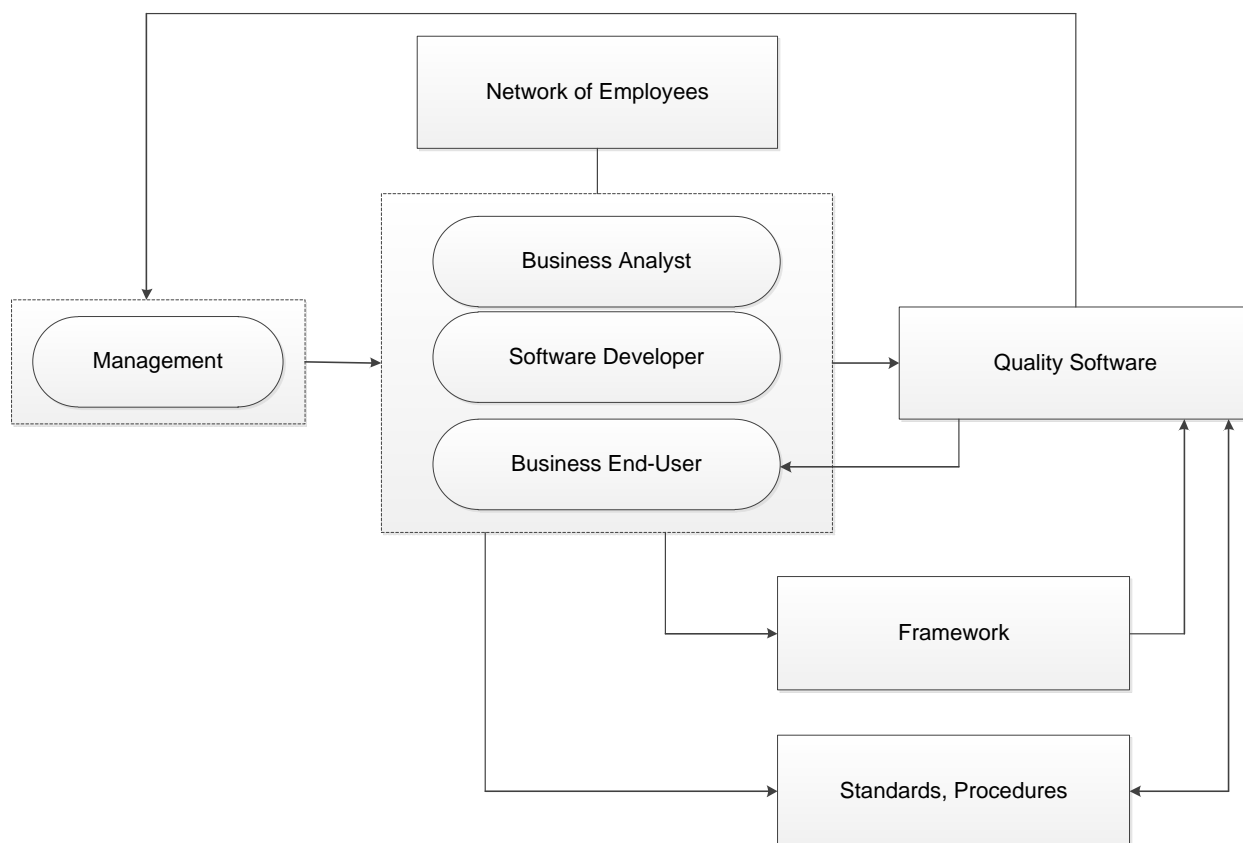


Figure 6.1: Factors influencing software testing and evaluation

6.2.1 Lack of testing framework

A framework can be regarded as a helpful structural tool that is often used to guide scope, boundaries and procedural activities. At Mootledi, however, there was no framework adopted and used for testing and evaluation of software. As a result, software was tested by different employees, such as a software developer, business analyst or end user, at various stages of testing. This was as a result of two factors: (1) individuals were selected to carry out testing based on favouritism, who the managers were more comfortable with, rather than on merit; and (2) employees were selected or rotated based on their availability.

Some of the implications arising from lack of testing framework were as follows: (1) there was no handover from one employee to another for smooth continuance of the process until completion; (2) there were inconsistencies in how testing was conducted, which detrimentally impacted the quality of some software in the organisation; (3) software testing and evaluation sometimes took longer to complete because employees might have been testing the same scenarios over and over without realising that these were tested before; (5) defects in software were hardly traced or tracked; and finally, (5) some crucial scenarios were missed as a result of poor structure, improper processes and many employees testing the same software.

6.2.2 Lack of management buy-in

Even though software was heavily relied upon at Mootledi Logistics for business processes and services, management buy-in was a challenge. There was no investment made in the testing and evaluation of software from the management perspective. For example, instead of purchasing proprietary software, they resorted to adopting open source software. This type of software does not require licensing for use and can be accessed for free. While the software development team trained themselves on how to use those tools, this sometimes had a negative impact on software testing timelines. The time spent, learning how to use the tools could have been utilised for actual software testing and evaluating activities.

Also, the organisation did not have a dedicated software testing team to perform testing and evaluation activities of software that were developed within the organisation. Software testing is a specialised field and requires a dedicated team. Software testers have to be trained and equipped with the necessary skills for testing various types of software at any given time. The lack of management buy-in and weak commitment did not entice many of the employees to commit to the concept of testing either. This frequently impacted the quality of software that was developed and managed in the organisation.

6.2.3 Network of employees

As in many environments, networks were consciously and sometimes unconsciously formed within Mootledi Logistics. During testing and evaluation of software, many of the employees worked along organisational structure, an indication of consciousness. However, some of the employees, specifically software developers, were uneasy with the formal approach, and rather preferred their individual approaches. Thus, some of the employees unconsciously formed the network in carrying out testing and evaluation of software in the environment.

Rather than follow the organisational structure, some employees identified themselves through interaction during the testing and evaluation of software. As result, some employees felt comfortable only working with certain other employees. Therefore, after software development, software developers would hand over the software to the business analyst or business end user with whom they felt comfortable working to test and evaluate the software.

That became the normative practice by some employees to determine who would test which software. However, the implication of performing testing and evaluation of

software in that fashion compromised the quality of software. If the business analyst or end-user was a friend to a software developer, they would be lenient in testing and evaluating the software, thereby imploring favouritism. As a result, software was not always properly or thoroughly tested. Those who were strict and thorough in testing and evaluating software were often bypassed or avoided because they were capable of detecting defects, which some software developers were uncomfortable with, in that it exposed their inability to do a good work.

In addition to friendship, some employees were selected to carry out testing of software based on availability. Such an approach enacted interaction and relationships, through networks that were unconsciously formed. Based on this network, some of the employees consciously and frequently made themselves available and were often selected for this reason as opposed to skill-set and ability. The impact of choosing employees on availability resulted in producing poor quality software. And poor quality software not only reflects bad on the IT department but had a negative impact on the business as a whole. When business is interrupted, this impacts customers because they couldn't reserve vehicles online, so revenue dropped and the image of the organisation got tarnished.

6.2.4 Quality of software

While the organisations preached quality at all times, this did not necessarily and often reflect on the software produced in the environment. *Quality of software* was challenged in that there were no real or formal criteria or requirements that could be used to guide the testing. The lack of framework and weak commitment from the management negatively affected the quality of software. Employees needed to be guided by some kind of framework to deliver quality software. Without such a framework, employees wouldn't be aware of what to cover when testing and evaluating software. The management support regarding software testing and evaluation was critical in the delivery of quality software. Management needed to invest money in software testing by establishing a dedicated software testing team, training software testers and purchasing proper software testing tools. The dedicated software testing team would have assisted the software development team in delivering quality software. Therefore, lack of commitment from management impacted the software quality.

Poor quality software stirred dissatisfaction from the customers. Customers couldn't receive the products and services rendered to them, services including booking of vehicles online. As a result, end users as well as customers couldn't perform their

duties because of dysfunctional software. While organisations believe in keeping their customers happy at all times, with poor quality software, it is impossible to achieve that.

6.2.5 Lack of standards and procedures

Standards and procedures set the criteria that can be used in the selection of software testing methods. There were various types of software testing methods that can be followed in conducting software testing and evaluation, including black box, white box and grey box testing. Black box testing is conducted purely based on the requirement specification knowledge. White box testing is conducted when the software tester has exceptional knowledge about the software as opposed to its functionality. Software testers are provided access to the code because they have the skills to perform testing from the code. Grey box testing, is conducted when the software tester has the limited knowledge of the software.

Standards and procedures also assist in selecting software testing tools and how to utilise those tools. It is a wasteful expenditure to purchase software testing tools and not know how to use them. Therefore, it is the responsibility of the management team to ensure that employees are trained and equipped with software testing skills as well as knowledge for using those tools. Software testing tools are expensive but when the organisation has those tools, they can save excessive time and money in the long run. These tools enable employees to create test requirements, test cases, execute test cases, and log defects. Reports for execution status can also be prepared from these tools.

Defects detected need to be fixed and retested, particularly as the fixing of defects may affect the functionality that was working previously. Therefore, after retesting the repaired defect, it is critical to perform regression testing to ensure that what has been fixed did not affect the functionality that was previously acceptable. The automation scripts prepared through automation testing tools could come handy in performing regression testing.

6.3 Findings and discussions: mmuso technologies

Based on the analysis of the empirical data from case 2, Mmuso Technologies, six factors were found to be critical to building a decision support system for testing and evaluating software in an organisation. As shown in Figure 6.2, the factors include software evaluation, process oriented, implementation policy, change management,

power relationship and organisational structure. These factors are presented in Figure 6.2 below:

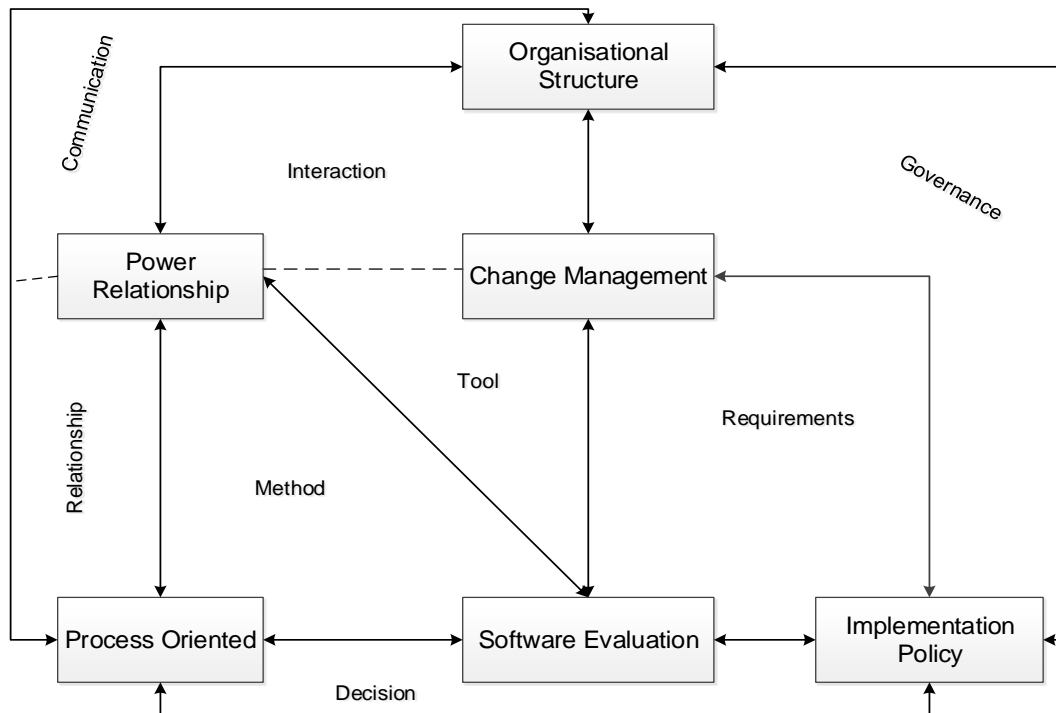


Figure 6.2: Factors influencing software testing and evaluation

6.3.1 Software evaluation

At Mmuso Technologies, software was evaluated from various perspectives such as sustainability, maintainability, usability and functionality. The focus on software sustainability was to evaluate whether the software can continue to be available for future purposes, such as its compatibility with newer platforms and changing business and technical requirements. The emphasis was on continuity mainly because some software gets discontinued after a short period of time. As a consequence, it become a loss to the organisation from a return on investment (ROI) viewpoint. Also, the software was tested for maintainability in order to ensure the ease in which the software can be enhanced for additional requirements, enhancement to correct faults detected, and adaptability to change of environments to improve performance. Organisations want to have flexible software that could be modified as change happens within the organisation.

Therefore, all software, new or with enhancements, needs to be thoroughly tested and evaluated to ensure it fulfils the requirements. This is an action that requires decisions at various levels, including business (product owners), management (product sponsor) and IT (technical experts). At Mmuso Technologies, a committee was responsible for accepting requests to develop or modify software for government

departments. Once the development or modifications were completed, the software testing team had to thoroughly test and evaluate the software, employing various testing methods and tools to perform the testing. Through those methods and tools, test results were produced to report on the software testing status. Other teams, such as the change and release management team, relied on those test results to make informed decisions regarding implementation of the software.

These test results included the test cycles performed during testing, the total number of test cases executed, and the total number of defects detected and resolved. Moreover, the test results included recommendations from the software testing team about the software tested and evaluated. That information assisted the change management team to make appropriate decisions as the test result determined whether or not the software was ready for deployment. The change management adopted implementation policy to deploy the software. However, there were situations where the change management was bypassed, which could have been associated to power relationships that existed. This often caused software failure, which resulted in immediate damage control that the organisation was always not prepared to carry out.

Once the software has been deployed, the management used their power to encourage end users to use the software.. It was not obvious that when the software was deployed within the organisation, end users would automatically adopt it. Hence, the management had to use the power inherent in their positions to diffuse the new software or the enhancement to the government department that requested the software. It was critical that the software function as expected as the potential of dysfunctional software provided a reason for end users to reject the software. The process of testing and evaluating software was quite intense.

6.3.2 Process oriented

As empirically revealed, software testing is *process-oriented* in that it follows a set of steps, instructions, guidelines and policies to complete, with the intention of producing quality software. Without such intensive and rigorous processes, quality would be difficult to achieve. The testing of software at Mmuso Technologies was not always automated; manual processes were also involved. Software testers had to execute a set of test cases manually to ensure that the software functioned as expected. Manual testing was a tedious process, repetitive and requiring full concentration of the tester. One of the challenges was that, at times, software testers were not allocated enough time to properly carry out their tasks. This resulted in software

testers working long hours, occasionally causing fatigue, which then detrimentally affected the quality of their testing – a poor cycle. Therefore, some test cases might mistakenly pass or fail. However, software testing automation is required to fast track the testing process by using the necessary tools. Currently, Mmuso Technologies make use of manual tools to capture test requirements, test cases and defects (IBM Rational Tools) as well as the performance testing tool (JMeter). These tools and processes require a decision support system that can seamlessly enable and support it.

The process includes the following: defects detected during the testing and evaluation of software which were logged and needed to be fixed by software developers. However, at times these defects erupted conflicts between the software developer and software tester who logged the defect. Such conflicts manifest from the relation that they have, a relationship of power. For example, the software developer would inform the software tester that the defect logged was not a defect. These conflicts tarnished the relationship between the two parties, who saw themselves as networks with the implication being that the quality of software was worsened.

Software testers needed to be analytical, innovative and able to communicate appropriately. Analytical skills enabled them to understand both the business requirements specification and the other group (network) of people that contribute to complete the tasks of software so that it can be implemented and used (diffused). The technical experts (software testers and developers) need to be more innovative to improve on the issues and factors that were not explicitly stated on the business requirement specification. This can be achieved through a decision support system that can validate various steps and required actions.

Communication also played a vital role because software testers needed to be able to communicate well with software developers in terms of building a relationship which can bridge an understanding of defects. Failure in understanding the business requirements meant that software testers would have neglected other scenarios unintentionally and the software would only be partially tested, potentially hindering the quality of the product. Software testers also needed to understand the environment in which the software was deployed. Software fails not only due to functional requirements but can fail due to non-functional requirements.

6.3.3 Implementation policy

Software testing and evaluation are intensive types of processes, which many employees in the organisation struggled to comprehend and abide by. Thus, an implementation policy is required, which was not in existence at Mmuso Technologies as revealed from the empirical evidence. Many of the employees recognise the significance of such policy. The management buy-in was necessary to enforce the necessary measures in implementing policy for software testing and evaluation. The software should be tested and evaluated through this intensive process for quality purposes. Necessary software testing processes must be followed as bypassing these processes affected the quality of the software.

It was necessary for the management to offer their support in terms of enforcing software testing processes between the project stakeholders. The software testing team has to abide by processes when testing software. Both the management and technical teams have to ensure the implementation policies are followed in the processes of development, testing and evaluation of software in the organisation. This can be done through governance, forming part of a system for decision support of the entire software production cycle.

This approach can enable requirements to be fulfilled, and as well prevent software from being deployed without the knowledge of testing and evaluation criteria and processes, which impact how change is currently managed in the organisation. For example, there were instances whereby software testing was bypassed, and yet the software still found its way to production. As a result, the software failed and the same team that was earlier bypassed was then requested to assist with the testing of that software. The rationale for not observing the implementation policies leads to bad quality software being implemented in production.

6.3.4 Change management

Change management can be linked back to the implementation policy. Any new software or enhancements to existing software were managed through the change management process. This was to ensure that the software functions as expected and meets the business requirements. Proper governance needed to be enforced by the change management team to ensure that only quality software was deployed to production within the organisation. Change management rely on the test results that are provided by the software testing team to make informed decision about implementing the software.

The change management can be manual or automated, but through a system that supports the entire process of software testing and evaluation. The change management includes completion and signed off documentation such as specification, change and closure reports. Other signed off documents, such as test plan and closure reports, serve as the entry criteria to the change management team stating the outcome of software testing. As a result, decisions are influenced and guided by the test results. Thereafter, the implementation policy can be adopted (or diffused) to allow the enhancement or new software to be deployed to production. This process assured the management that the software deployed can be of high quality and would not have a negative impact on existing software within the government department that requested for it. Bypassing the necessary processes can lead to implementation of poor quality software in the organisation.

6.3.5 Power relationship

In the software development project, various stakeholders, including project managers, business analysts, software developers, software testers and functional support personnel were involved at Mmuso Technologies. In the process of development, testing and evaluation of software the stakeholders interacted, through which relationships were created both consciously and unconsciously. The conscious relationships were often created through organisational structure. For example, software developers physically communicate to resolve logged defects. Unconscious relationships were guided by informal friendships and favouritisms (e.g. some software testers communicated with those with whom they were comfortable rather than what the structure of the organisation dictated in the deployment of software into production).

However, relationships could be twofold, good or bad, and often used as source of power. Good relationships are earned through respect, effective communication, and delivering tasks assigned to project stakeholders on time. This kind of relationship needs to be maintained as it motivates project stakeholders to perform their tasks to the best of their ability. As a result, quality software could be delivered before or on time. Good relationships also enable the project stakeholders to abide by software testing processes and policies to deliver quality software. Process-oriented implementation policy as well as change management played a vital role in managing power relationships, in that organisation objectives took precedence over individual preferences. When power relationships are managed well, it leads to conducive communication within an environment which enacts improved productivity of software delivery.

The management of power relationships can be implemented through an automated system that can support the entire testing and evaluation of software within an organisation. Otherwise, power relationships can also lead to bullying within workplace. As revealed from the analysis, some software developers bullied software testers over the defects they detected in the software. For instance, the software developer informing the software tester that they are unable to develop the software, the only thing they knew was to test it, whereas the software developer can develop and test. That resulted in a bad relationship that came about through disrespect and undermining roles of other project stakeholders. Such a relationship rendered the team dysfunctional and poor-quality software was produced. This impacted the testing results negatively.

6.3.6 Organisational structure

The *organisational structure* is a hierarchical arrangement in terms of authority, communications, rights and duties of employees in an organisation. Some of the essentials of the organisational structure include the following, (1) power of each or group of individuals is controlled through formal communication along various levels in the organisation; (2) the activities of change management are governed within the organisation's aim and objectives; and (3) policies are implemented by using the governance. These essential factors can be enabled and supported through a decision support system to reduce complexity and conflict of interest.

The organisational structure therefore helps in many ways such as, to determine how roles, power and responsibilities are assigned; how roles and responsibilities are controlled and coordinated; and how information flows between the different levels of management in the testing, evaluation and deployment of software. Every employee within the organisation when employed was assigned roles and responsibilities, functions they were expected to perform. Such roles and responsibilities come with some sort of power, to manage subordinates and control activities during testing and evaluation of software. For example, the test manager was responsible for creating a test strategy, test plan and various reports. They also supervised software testers to ensure that they were doing what they were tasked to do and to deliver on time. The test manager was expected by superiors to report regularly on the testing status whenever required. Therefore, the test manager had the power to assign tasks to software testers at any given time and discipline them if they were not fulfilling those tasks. Also, the responsibility of the test manager was higher than of a tester.

When project stakeholders undermined processes due to power they hold, organisational structure could be the solution to that problem. The management needed to manage the situation so that the software could be tested and evaluated. The management needed to outline and make project stakeholders aware of the software testing processes and implementation policies, and then enforce them for testing the software. Unruly behaviour by some project stakeholders also needed to be minimised to ensure that such behaviour did not undermine and disrespect other team members. The organisational structure needed to align processes, policies, changes and power to ensure that the testing of software is performed accordingly.

The project stakeholders needed to understand the effect that software testing could have when not performed. Defects would be detected in production by end users. As a result, IT department would appear as people who did not know what they were doing. As a ripple effect, management would be blamed for failing to do their work. Therefore, the organisational structure helped align processes in order for the project stakeholders to function properly.

6.4 Findings and discussions: bokamoso solutions

Based on the analysis of the empirical data from case 3, Bokamoso Solutions, six factors were found to influence testing and evaluation of software in an organisation as depicted in Figure 6.3. This includes, heterogeneity of testers, outsourcing, documentation, queuing system, standardisation and procedural. The factors are discussed below:

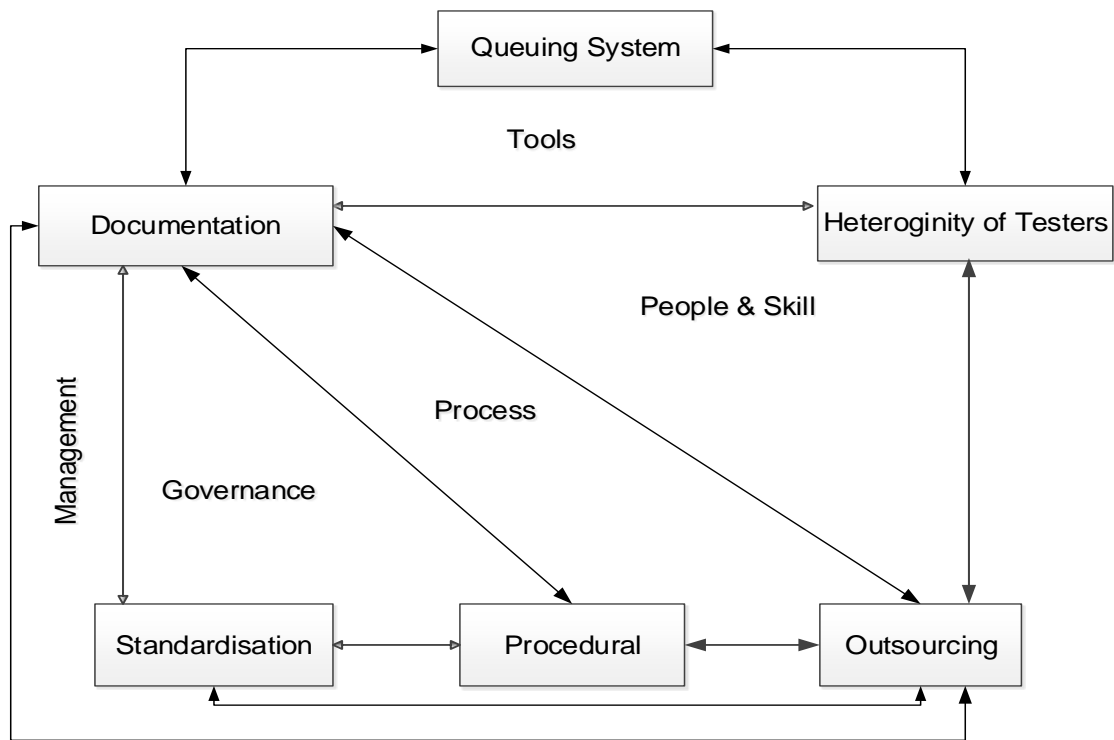


Figure 6.3: Factors influencing software testing and evaluation

6.4.1 Heterogeneity of software testers

Heterogeneity signifies diversity. In software testing and evaluation, software testers form part of the software testing team, a network. Software testers also form part of the software development team which is organised to develop, test and evaluate a particular software. As a result, teams can be part of other teams which make software testers and software testing teams heterogeneous. At Bokamoso Solutions, software testers belonged to the software testing team and project software development team. Multiple projects were tested and evaluated. As a result, software testers were involved in the software projects to which they were assigned. If assistance was needed in other software projects, software testers simultaneously worked on the software projects assigned to them as well as other software projects that required assistance.

At Bokamoso Solutions, there were also instances whereby some software projects were put on hold and software testers were temporarily reassigned to other software projects to assist with software testing and evaluation. Once the software project resumed, the software testers went back to work on their initial software project as well as the other software projects they were moved to. As a result, of this shifting, of software testers became heterogeneous. The value of heterogeneous software testers was that they were able to work in multiple software projects to produce quality software. Software testers were able to multi-task in multiple software projects

and perform various software testing activities to meet project timelines. Software testers who were involved in these software projects were both contractors and permanent employees. With their software testing skills, they made use of various documentations such as business requirements and technical requirement specifications to understand other software projects. Software testers needed to have the knowledge and understanding of the software that needed testing and evaluation.

6.4.2 Outsourcing

Outsourcing is the use of software testers from service providers who render software testing services to the organisation for a contracted period of time. Organisations outsource major functions to specialised and efficient service providers who ultimately become valued business partners. Bokamoso Solutions is one of those organisations that provides other companies with resources, specialising in various fields such as business analysis, software development and software testing. However, these resources needed to be highly skilled and knowledgeable in software testing and in making use of the software testing tools. Organisations that outsource software testing want resources immediately ready to perform software testing, not those who only wanted to start learning how to test or use software testing tools. They expect productivity from the outsourced resources from day one at work.

It was imperative for the organisation that was outsourcing software testing to have standards and procedures for conducting software testing. Those standards and procedures had to be there to guide outsourced resources in producing quality software. It was the management responsibility at Bokamoso Solutions to enforce those standards, as these standards had to be adopted irrespective of whether the software testers were internal or external. External resources were employed for a period of time to conduct software testing and evaluation, so it was important to manage them well in order to receive quality work. If standards and procedures were bypassed during software testing, the organisation would suffer because the quality of work received would not be acceptable. As a result, they would have lost their investment in software testing.

It would then be up to the internal resources to fix what had gone awry resulting in loss on the organisational side because they would have to pay their internal resources. Internal resources would then have to deal with the mess left behind by external resources. At the end of the day, the organisation would have lost money spent on software testing that was improperly conducted. So it was imperative for the management to ensure that the work of external resources was governed and verified

at all times to ensure that testing was performed according to organisational standards and procedures.

6.4.3 Documentation

Organisations document strategies, policies, standards and procedures so that they could be shared. Any new employee could access any of those documents to learn how the organisation functions. Even in software testing, the same principle applies. Organisations compile documents such as test strategies and test policies at the organisational level. The test strategy outlines the testing approach and informs the project stakeholders about some key issues surrounding the testing process. The test policies provide the direction which the testing team should adhere to. Every organisation operates differently, each having a unique way of doing things. Bokamoso Solutions had to adhere to their partner's test strategies and policies for performing software testing and evaluation.

Documentation in software testing plays a vital role in testing because software testers rely on it to plan and execute their software testing activities. The documentation needed from the beginning to the end of software specified the requirements. Out of the requirements, the test plan was compiled by the software testing to outline the scope of testing. Also, testing activities such as test requirements and test cases were extracted from the requirement specification. The test cases covered scenarios that needed to be tested and evaluated.

During and at the completion of software testing, reports were produced to retain the testing status. Test managers produced these reports for reporting purposes to the senior software test manager. The test results from these reports were also used for software evaluation prior to its deployment. Every process and activity that happens in software testing needs to be documented because software is not only tested once. Whenever enhancements were made to the existing software, it needed to be retested. Therefore, some of the documented information, if not all, could be re-used to re-test the software. Updates to that information would be retained to be re-used. As software testing is a repetitive process, whenever software is enhanced or upgraded it needed to be re-tested. This information equipped the software testing team with the awareness of what needed to be done and how to do it. It also assisted with upskilling and knowledge transfer between software testers.

6.4.4 Queuing system

As stated earlier, a list of software projects that needed to be tested was issued to the senior test manager. As such, a *queuing system* was required to prioritise the projects within the organisation. When projects were not in a queue, the organisation was running a risk of not fairly assigning projects for testing, imposing a challenge of suspending projects in the middle of testing at Bokamoso Solutions' partner. Consequently, it was necessary for the organisation to introduce a queuing system that would serve to prioritise its projects. Fairness in queuing projects was not to be underestimated and it would assist in properly prioritising projects requiring testing.

This system eliminates situations whereby projects are prioritised through favouritism. If the product owner or end user was not liked by the senior test manager their projects were shifted last or not even tested at all. The queuing system was needed to allocate projects fairly.

6.4.5 Standardisation

Every software that was developed or enhanced needed to be tested and evaluated following a particular standard agreed upon within the organisation. The organisation that Bokamoso Solutions was providing testing services for had their own testing. As a result, Bokamoso Solutions had to adhere to those standards in delivering quality software. Standards were twofold: internal and external. Internal standards were agreed upon within the organisation, implemented using templates. For example, the test manager had to use templates to create a test plan, test closure reports and test estimates. Deviations from standards were possible although not recommended and require a thorough motivation that was reviewed and signed off by the custodians of the standards within the organisation. Standards might include but are not limited to naming conventions, test case design and coding standards for automation testing.

External standards could be international such International Organisation Standard (ISO) and best practices in testing software. Some of the organisations for which Bokamoso Solutions was conducting software testing and evaluation were audited from time to time to ensure that the software complied with these international standards. Many organisations around the globe develop and implement different standards to improve the quality needs of their software.

By implementing these standards, the organisation was adopting the internationally-recognised and agreed standards for software testing which provided the organisation with a high-quality approach to testing that could be communicated throughout the world. The rationale for testing the software according to the

international standards enabled organisations to compete globally with their quality software. Any organisation that was not compliant to international standards ran a risk of losing their quality certificate and having their name removed from ISO. Organisations that were compliant with international standards were still able to compete internationally.

6.4.6 Procedural

The entire software testing was *procedural*, meaning that software testing activities were procedural. For example, software cannot be tested if the business requirements were non-existent because the developed software needed to be verified against the business requirements. The test results could not exist if the software was not tested. Therefore, sequence was vital in testing and evaluating software. The business requirement specification was necessary for the test processes and activities to be created (such as testing strategy, test plan, test design and test execution).

To achieve proper software testing, the above-mentioned testing processes and activities needed to be followed. When testing and evaluating software, Bokamoso Solutions did its best to comply to the best practices as well as ISO standards for producing quality software that could be used to compete globally, as some of its customers were ISO registered. The standards and the best software testing practices aided the organisations in eradicating the worst incidents that can occur while software was in production.

6.5 Decision support system framework for testing and evaluating software

From the findings, seven factors were found to have a critical influence on the testing and evaluation of software within an organisation: requirements, methodology, filtering, repository, governance, assessment and institutionalisation. Figure 6.4 depicts these factors and how they relate to each other. Each of the factors consist of phases (P1 . . .Pⁿ) in the sequential order of the activities. Some of the activities can be implemented in parallel. For example, policy, standard and principles within governance can be carried out concurrently. To understand the framework, the discussion should be read in conjunction with Figure 6.4. These factors are discussed below:

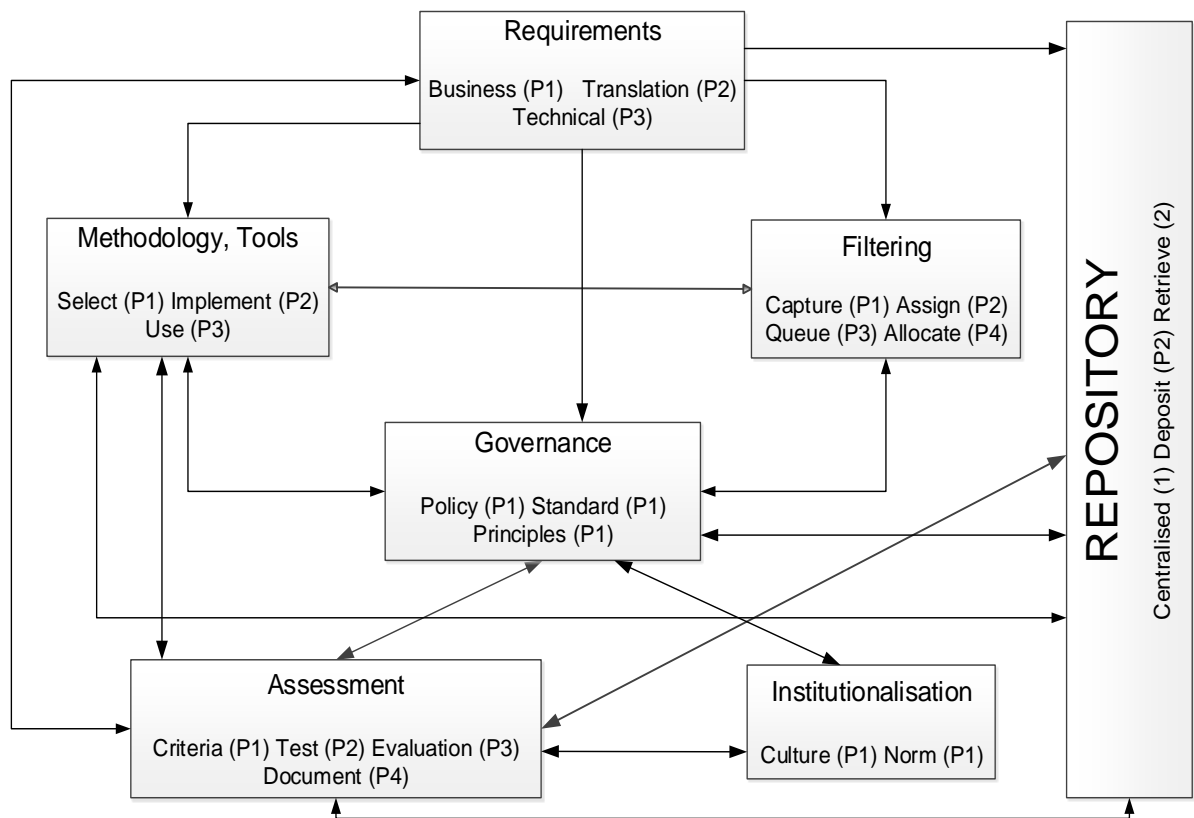


Figure 6.4: Decision support system framework for testing and evaluating software

6.5.1 Requirements

In software testing and evaluation, *requirements* could be either functional or non-functional, coming from both business and technical units, of an organisation respectively. The functional requirements basically describe what the software should do. Some of the typical functional requirements include factors such as business rules, authentication, external interfaces, reporting and administrative functions. The non-functional requirements describe how the software should technically behave within the environment. Non-functional requirements cover all the remaining requirements not covered by the functional requirements. The non-functional requirements specify the criteria used for the assessment of software in an organisation. For example, the software should be able to coexist with other software in the environment. Also, the non-functional requirements elaborate a performance characteristic of the software. Some characteristics of non-functional requirements include response times, throughput and utilisation of the software.

Manual software testers and automation testers extract the functional requirements from the business requirements they receive from the business analysis team. These functional requirements enable the creation of test requirements and test cases which are then captured in the software testing tools adopted by the organisation. According to Hooda and Chhillar (2015), a test case outlines the steps required to test any

functionality of the software and contains expected and actual result. Test cases are basically scenarios that have been identified from the requirement specification. These test cases can be automated by automation testers to assist the manual testers with regression testing. Therefore, organisations can purchase software testing tools or utilise open source tools, depending on the investment the organisation is willing to make in software testing.

Performance testers make use of non-functional requirements to test the performance of the software, monitoring the software's continuous load on the network. For example, potential leaks can be detected in memory utilisation along with analysis of performance degradation and how the software copes under strenuous use. Performance testing is also performed through software testing tools adopted by the organisation. Testing concurrent authentication, for example, means that a hall would be filled with a number of users that need to log concurrently on the software tested. As it would be too expensive to get, for example, 1000 users in the same room to log in concurrently on the particular software at same time. Also, a database nowadays lock inputs into the database for creating records per user. Therefore, it queues the inputs and create individual records in the database.

Two good examples of software that failed in production because performance testing was not conducted are school online registration in Gauteng and the Fly Safair website. Functionally of the software was working as expected but failed when concurrent users logged into this software. "The minister of education in Gauteng Panyasa Lesufi said the software crashed after it received 600 hits per second. The software was upgraded to receive 3 000 hits a second but that still failed and was increased again to 20 000 hits a second" (Monama *et al.*, 2016:n). "The Safair Airline experienced high volumes of sales due to its R1 birthday reduced on flight tickets. The company says it managed to sell at least 5,000 tickets so far and will extend the deadline for the sale" (Koza, 2015:n). Performance testing is crucial because if not performed, organisations experience a loss or fail to achieve objectives.

Both functional and non-functional requirements get stored in the software testing tool used by the organisation. Some organisations that don't have these tools, create test cases on spread sheets and store them on a repository such as Microsoft share point, hyper wave, intranet or a shared drive. This storage enables any other team that might need that information to easily access or retrieve it.

6.5.2 Methodology

Software testing *methodologies* are the different ways of ensuring that the software under test is fully tested. Methodologies and tools are selected and implemented based on organisational requirements. Software testing methodologies encompass functional and non-functional testing to validate the software under test. The testing methods include unit testing, integration testing, system testing and performance testing. As software increases in complexity and enmeshed with the large number of different platforms and devices that need to be tested, it is more important than ever to have robust testing methodologies for making sure that software being developed has been carefully tested. This is to make sure that the software meets its specified requirements and can successfully operate in all anticipated environments with the required usability and security.

The software testing methodology has a definite test objective, test strategy and deliverables. Irrespective of which software development methodology (traditional or agile) has been adopted within the organisation, the testing methodologies stated above can be applied in the testing and evaluation of software depending on the scope of the project. However, to successfully apply some of these methodologies software testers require software testing tools. For example, performance testing is performed to determine the performance of the software on the network in terms of response, speed and stability under a particular workload.

Both human and non-human actors are dependent on each other to test and evaluate software. This is to ensure that quality software is delivered to those who requested it. Software testers, including manual, automation and performance, require software testing tools to perform various types of software testing, as stated above. These tools could be proprietary or open sources. According to Sharmila and Ramadevi (2014), Load Runner is a tool that can be used in determining the performance and outcome of the software under load. Software testing tools enable software testers to successfully perform their duties.

6.5.3 Filtering

Filtering in software testing and evaluation is a process of removing unwanted functionality or defects from the software, a process guided by requirements as illustrated in the framework (Figure 6.4). The main activities of the process is as follows: (1) capture the software into the systems; (2) thereafter the software is assigned to a domain; (3) this is put in a queue; (4) and then it is allocated to personnel for testing and evaluation. The fact is that human beings are prone to

making mistakes. For example, the business analysts can incorrectly state the business rule in the requirement specification or specify a requirement ambiguously. If the mistake is not picked up by the software developer, the business rule will be built into the code and the software wouldn't behave as expected. When software testers are testing the software, the incorrect business rule would be detected as a defect because the software would not be functioning as expected. Therefore, detecting such defects is filtering unwanted functionality from the software. Regarding ambiguous requirements, the software tester would not be in a position to create some test cases due to unclear requirements. As a result, clarity regarding those requirements would be required from the business analyst. This, then, is a filtering process in terms of requirements.

When compiling test plans, the test managers are able to identify risks from the requirement specification. Also, during execution of test cases, risks could be identified which might impact detrimentally on the quality of software. These risks must be mitigated to produce quality software. Defects detected and logged during the testing and evaluation of software form part of the filtering process and it is the responsibility of the software testing team to filter all unwanted things from the software under test in their efforts to deliver quality software.

6.5.4 Repository

A *repository* generally refers to a central place where information gets stored, accessed and maintained. The repository is defined by the organisational requirements. Activities of governance and assessment, including the methodology and tools that are applied for testing and evaluation are stored in the repository, primarily to enable and support the ease of access to the stock of organisational knowledge, which fosters quality testing and evaluation of software. Additionally, a repository enables control of organisational stock. Thus, those who wish to access the information that is stored in the repository must apply for access to retrieve whatever information they seek. All materials or information stored in the repository must be secured at all times to protect organisational information against attack and leakages.

In terms of software testing and evaluation, requirements which could either be functional and non-functional must also be stored in the repository so that when the project stakeholders are in need of this information, they can easily access it. In some organisations, teams that are involved in software testing and evaluation are not

situated at the same premises. Therefore, when information is stored in the repository, it is easier for other project team members to access such information.

There are other documents prepared by the software testing team which include statements of work, test plans and test closure reports which also must be stored in the repository as auditors require these documents during auditing to validate how the organisation has tested and evaluated their software. This information assists them in compiling their audit reports for the organisation they are auditing. Also, the information that resides in the software testing tools such as test cases, execution of those test cases, and defects logged and fixed for the particular software assists auditors in validating whether or not the software was tested and evaluated according to software testing standards. All these audit findings aid the organisation in fixing their mistakes and improving how they test and evaluate software.

6.5.5 Governance

Governance plays a pivotal role in the process of evaluating the quality of the software. Governance includes policy, standards and principles which can be applied in the process of software testing and evaluation concurrently. *Governance* involves the definition of organisational test processes, test documentation and the derivation to testing techniques. The software testing process affords the organisation with governance on ways to implement the adopted testing policy, standards and principles that aid the stakeholders to deliver quality software. Test governance enforces compliance to the organisation's testing process. Governance also ensures that the testing processes are continually improved to ensure the constant, uninterrupted delivery of quality software.

Key documentation such as the test policy and the organisational test strategy requires management support as these documents form part of the organisational test processes. According to IEEE 29119, testing processes can be broken into three parts: organisational test processes, test management processes and dynamic test processes. All three test processes have key documentation that goes along with them for a successful testing organisation. The test policy, for example, defines the overall principles that guide testing in the organisation. This document is the primary testing document informing the entire testing organisation of why software testing is performed. All other testing documentation and processes are based on the test policy. It should include the test policy statement, policy principles and testing standards, as it is the foundation provision of the testing processes (IEEE 29119).

There are various standards available in the testing industry. The common one is the IEEE 29119 which has a series of five standards. The purpose of the standards is to provide generally acceptable methods of testing across the entire testing industry. Organisations can choose to comply fully with these standards or opt for partial compliance. The standards are not only limited to the international standards, the organisation can still produce and implement internal testing standards that can best fit the local context of their organisation.

The principles of software testing, regarded as the beliefs of the testing organisation, remind the software testers within the organisation the reason why they test and how they test in the first place. The principles form the foundation of the software testing organisation and are clearly articulated in the testing policy. These generally accepted testing principles are formulated by testing industry bodies like the International Software Testing Qualifications Board (ISTQB). These principles form part of the early training and development curriculum in software testing.

All these policies, standards and principles of software testing must be stored in the repository for the software testers to access to remind themselves of the best practices.

6.5.6 Assessment

Assessment promotes quality of software in an environment. Thus, many organisations find value in benchmarking their progress to improve their processes through assessment. Organisations that develop or enhance existing software have test processes in place to test and evaluate their software. The software testing processes begin with test planning, designing of test cases, preparing for execution and evaluating status till the test closure. During the test planning, the scope and risks, test approaches and testing objectives are identified, enabling the software testing team to identify how much testing needs to happen and what possible risks might be encountered.

The next stage is the analysis and design where software testers identify test conditions, evaluate the testability of requirements and the test environment is set up. During the test implementation, test cases are prioritised, and test data is created for those test cases. Thereafter, the test cases are executed. Once the execution is complete, the software testing team reports on the outcome of testing and the test closure report is compiled. Organisations that test and evaluate software have their own testing processes in place.

Therefore, organisations must assess their software testing process to improve the way in which they conduct software testing. It provides the opportunity for the organisation to know itself and its competition better. As a result, the organisation can strive to produce quality software that would enable it to out-compete its rivals. Through requirements, organisations are able to identify testing objectives; they are able to determine the scope of testing, determine which testing approaches to employ and determine risks that might be incurred during testing. All these activities are documented and stored in the repository for future reference.

6.5.7 Institutionalisation

Institutionalisation is a state of stability that is required in software to guarantee quality. It is ways, such as continuous assessment and adherence to governance, in which the software team performs their daily activities for testing and evaluation of software. Iyamu (2011) defines *institutionalisation* as the process where practices are assimilated into the norm: the ways in which project stakeholders perform their testing and evaluation of software, finally becomes the organisational norm. Those norms should match the internationally agreed set of standards for software testing applicable within any organisation. By implementing these standards, the organisation would be adopting the only internationally-recognised and agreed standards for software testing, giving the organisation a high-quality approach to testing software.

Such norms become the software testing culture that is adopted by software testers within the organisation. The culture of software quality must be practised in all parts of the organisation because quality is essential for success. Software teams involve various stakeholders such as project managers, business analysts, software developers, software testers, designers, product owners and executive. All these stakeholders play a role in the quality of the final software. Because of this, they need to align their work and practices with agreed international standards, best practices and the test maturity levels in order to deliver quality software. Quality software enables the organisation to compete locally and globally with other organisations.

Finally, institutionalisation and norms manifest into the organisation culture. Organisational culture is a combined means of regulating the behaviour of employees within organisation which diffuses all activities as catalysts for the development and growth of the organisation (Gavric, Sormaz & Ilic, 2016). So if the foundation of this organisational culture is hinged upon incorrect norms, then the organisation would not

be in a position to produce quality software. Thus, organisations need to align themselves with the best software testing practices, ISO testing processes as well as testing maturity models. In so doing, the best organisational culture would be practiced by employees. The organisational culture must be documented and stored in the repository so that new employees joining the organisation can learn how quality software is produced within the organisation. Moreover, existing employees can remind themselves of certain practices they may have forgotten.

6.6 Summary

The findings for each case were explained separately and a diagram was created. The diagram created relates to the explanation of the findings. Lastly those findings were interpreted and the decision support system framework for testing and evaluating software within organisations was created. This framework could be adopted by either private, public or small medium enterprise for testing and evaluating software. This is due to the three cases which were employed for this study. The next chapter concludes the study, and proposes and recommendations.

CHAPTER SEVEN

CONCLUSIONS AND RECOMMENDATIONS

7.1 Introduction

This final chapter of the thesis presents the recommendations made and the conclusions drawn from the study. The research problem was driven by the need to understand the challenges occurring during software testing and evaluation in organisations. Literature was reviewed to escalate the understanding of the factors and elements which relate to the objectives of the study. The literature was collected from authentic academic databases. A qualitative research method, together with the case studies, was adopted for this study. Various methods, techniques and approaches were employed in the study to achieve the study objectives. Additional detail on the research methodology was covered in Chapter 3.

The study was underpinned by actor network theory (ANT) and diffusion of innovation (DOI). These two theories – actor network theory and diffusion of innovation – were applied as lenses at macro and micro levels in the data analysis. ANT was used at macro level to gain understanding of testing methods, selection of tools and factors influencing the testing and evaluation of software in organisations. DOI was used at the micro level, to diffuse the innovation. Three organisations were selected as case studies to investigate more thoroughly how software was tested and evaluated in different organisations. The cases consisted of a private business, a public business and a small and medium enterprise (SME).

Interviews were conducted separately, while documents were collected from the three organisations in the study. The purpose of conducting the interviews and collecting documents was to understand with more precision how software projects are tested and evaluated in the organisations. The interviews were conducted with various employees involved in the testing and evaluation of software projects in the organisations. The interviews were recorded, transcribed, and responses analysed with the moments of translation from the perspective of the actor network theory as well as the innovation decision process from diffusion of innovation theory. The findings from the analysis of the three case studies were interpreted separately and then based on the analyses and interpretations, a guiding framework was developed.

The objective of the framework was for proposing a decision support system to be used in understanding and addressing the challenges which occur during software

testing and evaluation in organisations. This included understanding how software testing tools were selected, methods involved in software testing and the factors influencing the testing and evaluating of software in organisations. The framework will guide the organisations in addressing the challenges which occur during software testing and evaluation because these challenges, if left unattended, have the potential to hamper software projects.

7.2 Summary of the study

The study is divided into seven chapters. The outline of the thesis is presented briefly in Chapter 1. The subsequent chapters are summarised as follows:

CHAPTER 1: Introduction

This chapter introduces the research topic as documented in the study while simultaneously provides the introduction to the full study: the research problem, research objectives and research questions. While previous studies have been conducted around software testing and evaluation, there is no single study that has yet developed a decision support system framework for addressing challenges encountered during testing and evaluation of software in organisations. The use of three organisations as case studies makes the case for generalisation (Lokke & Sorensen, 2014). The framework is intended to guide organisations in making informed decisions about testing the software as well as eradicating challenges that are encountered during software testing. Software testing is the most essential part of the software development life cycle. Failure in recognising this, can likely result in organisations implementing poor quality software.

This chapter also covers the literature review relating to the study, research methodologies which were applied and the two underpinning theories that were applied in the data analysis for assessing the findings and drawing conclusions. Thus, the chapter provides an overview of the entire study, specifically concerning the organisational structure of the thesis.

CHAPTER 2: Literature study

This chapter presents the discussion on literature related to the testing and evaluation of software within organisations. The literature review covers six main parts of the study, including software development, software testing, software testing methods, software testing tools, decision support system and the theoretical underpinnings of the study. The two theories underpinning the study, actor network theory (ANT) and diffusion of innovation (DOI), were also discussed. The moments of translation, also

known as the lens of ANT, was applied. Thereafter, the innovation decision process was applied to diffuse the innovation.

CHAPTER 3: Research methodology

The research methodologies used in the study were discussed in this chapter. The research methodologies include approaches, methods and techniques that were applied in the study. The researcher applied a qualitative research method. The case study and various data collection techniques such as interviews and documentation were also employed. The case study research approach was employed in the study and interviews conducted from which to collect data from all three case studies (a private business, a public business, and an SME). The intention was to understand how these three organisations conduct software testing and evaluation with the intent of developing a decision support system framework for testing and evaluation of software in an organisation.

Semi-structured interview techniques were used to collect data, a technique enabling the researcher to record conversations, take notes and refine the research questions during the interviews process, in order to clarify things that were unclear during the interviewing process. Interviewees were assured that their anonymity would be respected and their right to privacy upheld. Also, organisational documents such as organisational structure, standards and policies about the organisation were gathered in addition to the information received from respondents. The interpretivism approach was employed in this study to explore participants' subjective views of their own experiences in their own environments regarding software testing and evaluation.

CHAPTER 4: Case study overview

The overviews of the three cases selected for this study are presented in this chapter. This includes the goals, strategy and vision of the individual organisation, the unique organisational structures, and the roles and responsibilities of the departments within the organisations. The three case study interviews were conducted using the same strategy, but the organisations were treated differently since they operate in different businesses (private, public and small medium enterprise). The businesses chosen as case studies are not in competition with each other and have entirely different cultural settings, one being in the automobile and logistics sector and others from the information technology (IT) sector.

CHAPTER 5: Data analysis and findings

The analysis and findings from the three case studies are presented in this chapter. The analysis were carried out using moments of translation from the perspective of the actor network theory and the innovation diffusion process from the perspective of diffusion of innovation, the two theories extensively discussed in Chapter 2. The stages of the moments of translation were used to analyse the data. Also, the innovation decision process stages were used to analyse data. Actor network theory was used to establish the relationships and examine the interaction between actors (human and non-human) as well networks involved in the testing and evaluation of software.

Diffusion of innovation was used to examine how tools and methods were diffused towards testing and evaluation of software in the organisation. This includes the knowledge that was gained for making decisions that persuaded other actors who participated or did not participate in the process of software testing and evaluation. The analysis of the three cases were done separately, but in the same format with insights from the analysis presented in this chapter.

CHAPTER 6: Findings and interpretation

The findings and interpretations of this study are presented in this chapter, with the findings for each case explained separately. Based on the findings and interpretation, the decision support system framework for testing and evaluating software was developed. The framework is aimed at addressing the challenges which occur during software testing and evaluation in organisations.

CHAPTER 7: Conclusion and recommendations

This is the last chapter of the thesis and it summarises all the chapters. The chapter provides the evaluation of the study. The theoretical contributions of the study are presented and the recommendations and suggestions for further research are also covered in this chapter.

7.3 Evaluation of the study

The conclusions have been drawn from the analysis of the data, findings from the analysis and interpretation of the findings. The findings became the results from the answers to the research questions. The research objectives of the study as stated and repeated in Chapters 1 and 3, respectively, include the following:

- i. examine and understand the tools (for manual, automation and performance testing) used for software testing;

- ii. explore and understand methods (approaches adopted for testing such as white or black box testing) involved in testing the software;
- iii. examine the factors (factors triggering testing to be conducted) that could influence the testing and evaluating of software in organisations; and
- iv. Create a decision support system framework based on the findings from the objectives as stated above. The aim of this decision support system framework is to address the challenges which occur during software testing and evaluation in organisations.

To achieve the above-mentioned research objectives, four main questions were formulated. Each of the questions had sub-questions during the data collection. The main questions include the following:

- i. What are the tools used in testing software?
- ii. What are the methods involved in the testing of the software?
- iii. What are the factors influencing the testing and evaluating of software in organisations?
- iv. How can a decision support system framework be developed and used in addressing the challenges which occur during software testing and evaluation in organisations?

In evaluating the study, the research questions, which follow, are discussed to ascertain how the objectives of the study were fulfilled:

i. What are the tools that are used in testing software?

Software testing can be performed manually or with software testing tools, tools which are either free, open source or proprietary software. Open source tools are often provided freely by those who developed them, freely downloaded from the Internet. Proprietary tools, on the other hand, are commercialised, which means that they require licensing right per use. The tools can be cost prohibitive, making affordability difficult for some organisations, from a purchase viewpoint. As a result, some organisations opt to using a Microsoft spread sheet to capture test requirements, test cases and defects logged. Other organisations, though, can afford to invest larger amounts of money in software testing. This investment is used to set up independent software testing teams and purchase software testing tools that enable them to more readily perform various types of testing.

However, there is no single software testing tool that allows an organisation to perform multiple testing activities such as manual, automation and performance testing. As a result of tool limitations, the organisation fails to achieve its objectives in

conducting end-to-end testing. Software testing needs to cover all requirements, both functional and non-functional. Therefore, the software testing team cannot claim to have produced quality software if they have only covered functional requirements or non-functional requirements as opposed to both. Consequently, organisations need to acquire these tools to cover the entire spectrum of software testing. For organisations it is costly, but for the companies producing these tools, it is profitable. The main reason for separating these tools is for the suppliers to make profits.

Having explored all three cases, one organisation opted purely for open source tools. However, they couldn't use Selenium effectively to perform automation testing because they relied on self-training. There wasn't enough time, therefore, for them to learn because they were also involved in the development, testing and evaluation of software. As there was no independent software testing team, there was lack of skill regarding software testing in this organisation. As a result, poor quality software was produced.

The second organisation purchased IBM rationale tools which enabled them to capture test requirements, test cases and log defects. Due to low budget, however, they adopted JMeter (open source) for performance testing. At least they had an independent software testing team, a manual testing tool and a performance testing tool. They had the advantage of producing quality software because they were able to cover both functional and non-functional testing. The only disadvantage was that they did not have an automation tool to accelerate the testing and evaluation of software. Therefore, they required too many testers to perform manual and functional testing.

The third organisation was using HP Quality Center (manual), Unified Functional Testing (automation) and LoadRunner (performance), proprietary tools produced by Hewlett Packard but sold separately. However, these tools can integrate to each other. For example, test cases in Quality Center can be executed manually and also through automation scripts created through Unified Functional Testing tool. These automation scripts can be linked to manual test cases and be run automatically within Quality Center. As a result, the third organisation had the advantage of producing quality software. However, it is possible to deliver poor quality software even if the organisation relies on these testing tools. Software testing is procedural; therefore, it is imperative for the software testing team to follow software testing standards and procedures. Bypassing them would negatively impact software quality.

ii. What are the methods involved in the testing of the software?

Software testing methods are the approaches that can be adopted by the software testers to test and evaluate software within the organisation. Thus, software testers are trained and equipped with necessary testing knowledge to assist with testing the software. These testing methods include black box, white box and grey box testing which have been discussed extensively in Chapter 2. Experienced software testers know which testing method to follow and when. For example, black box testing is performed when the software tester does not know the internal workings of the software. Those who know the internal workings of the software and have programming knowledge perform white box testing. Software testers with limited knowledge of the software conduct grey box testing.

Not anyone is qualified to be a software tester. Software testing is a career field just like software development, project management and business analysis. There are international software testing standards approved by ISO which need to be adopted to produce quality software. Software testing is process-oriented; therefore, software testers must follow testing processes and frameworks to deliver quality software. It is an intense process which requires software testers to carefully follow a set of steps, instructions, guidelines and policies to produce quality software. The lack of framework, lack of standards and lack of procedures within the organisation serve to compromise software quality. It is like picking up someone from the street who doesn't have a clue about software testing and simply instructing them to test the software.

iii. What factors influence the testing and evaluating of software in organisations?

Software is a product and therefore like every product released to the public or within the organisation, it needs to undergo testing. The software testing team needs to verify and validate whether the software behaves as expected. They need to test and evaluate both functional and non-functional requirements of the software. Performing functional testing enables software testers to detect defects which could be fixed while the software is still undergoing testing. Detecting defects in production is risky because it hinders business, it taints the image of the organisation and impacts customer reaction negatively. Quality software that sustains and enables the organisation to be competitive must be delivered to business. However, even if the software works as expected, this does not necessarily mean it will automatically function without testing the non-functional requirements. Performance testing is performed to determine the response and stability of the software under countless workloads, measuring the quality attributes of the software such as scalability,

reliability and resource usage. The Department of Education in Gauteng and the Safair Airlines previously encountered performance challenges whereby their software couldn't handle the load of users accessing their software. Therefore, it is vital for business to cover all the requirements when testing the software.

iv. How can a decision support system framework be developed and used to address the challenges occurring during software testing and evaluation in organisations?

Analysis and interpretation of the data indicate that if challenges that occurs during software testing and evaluation are not addressed, they will continue to impact the quality of software negatively. The consequences of not addressing these challenges will result in the software project not being implemented or, perhaps even more deleteriously, being implemented with defects. As a result, the organisation's challenges will negatively impact business and customers.

Using the lens of ANT and innovation decision process of DOI in the analysis, certain factors were found to clearly influence the testing and evaluation of software. Based on the interpretation of these factors, a framework was developed. The decision support system framework for testing and evaluating software was designed to assist in addressing the challenges occurring during the testing and evaluation of software in the organisation.

How the objectives of the study were achieved:

i. Examine and understand the tools (for manual, automation and performance testing) used for software testing

There was an evident lack of management buy-in within the organisation. It was evident that management was not willing to invest money in software testing. Firstly there was no independent software testing team to test and evaluate software in order to produce quality of software within the organisation. Employees who specialised in other fields such as business analysis and software development were tasked to do the software testing. As software testing is a specialised skill, the organisation needed to utilise trained software testers to perform software testing. Secondly, free open source software testing tools were adopted: instead of purchasing proprietary tools, the organisation settled for free open source software testing tools. Employees researched these tools, trained themselves on these tools and adopted those tools. However, they couldn't fully utilise the tools but used the tools only to perform automation testing. As a result, proper quality software couldn't be achieved. Hence, this research objective was achieved.

ii. Explore and understand methods (approaches adopted for testing such as white or black box testing) involved in testing the software

The following factors indicated that quality was not taken seriously: process-oriented, lack of framework, lack of standards and procedures, software evaluation. Software testing is process-oriented. It is an intense process which requires software testers to follow a particular set of steps, instructions, guidelines and policies to produce quality software. Therefore, if the organisation does not have a dedicated software testing team trained to perform software testing, they wouldn't know how to test, what to test and when to test what. Because software testing is procedural, software testers follow a particular sequence to execute their testing activities. Also, if there is no testing framework, no testing standards and no procedures, employees wouldn't know how to test and evaluate the software. As a result, quality software cannot be delivered. This objective was also achieved.

iii. Examine the factors (factors triggering testing to be conducted) that could influence the testing and evaluating of software in organisations

Organisations rely on software for competitiveness and sustainability. Therefore, organisations must continue to develop software while also enhancing existing software. It is evident that this creates a need and influences the software testing and evaluation in organisations. However, in order for the software testing team to be able to test, they require documentation such as business requirement specifications and technical design specifications. They must follow testing standards and procedures when conducting software testing. Both functional and non-functional requirements must be integrated to achieve quality software. Various teams interact and work together in order to deliver quality software. Therefore, this objective was achieved.

iv. Based on the findings from the objectives as stated above, a decision support system framework will be created. The aim of this decision support system framework will be to address the challenges occurring during software testing and evaluation in organisations

The decision support system framework for testing and evaluating software in organisations was achieved based on the findings from the three selected organisations. Therefore, any organisation either private, public, or small to medium may adopt this framework in testing and evaluating software. This framework, when followed, will guide the organisation in delivering quality software.

7.4 Contribution of the research

This section presents the contribution of the research from theoretical, methodological and practical perspectives.

7.4.1 Theoretical contribution

The study contributes to body of knowledge through its addition to literature. From this study, two articles have been published: (1) Diffusion of innovation theory for information technology decision making in organisational strategy; and (2) The connectedness in selecting socio-technical theory to underpin information systems studies.

(1) Diffusion of innovation theory for information technology decision making in organisational strategy

The Diffusion of innovation (DOI) theory was employed as a lens to examine the influencing factors and how decisions were made in applying technologies for organisational strategy. As a result, the IT decision-making framework for organisational strategy was developed to guide organisations in diffusing systems and technologies to enable organisational strategy. The framework outlines the organisational activities, technologies and governance. The organisation activities included culture, people and operations. Technology covered the dimensions of systems, innovation and adoption. Governance included transformation, awareness and collaboration. Therefore, IT decisions needed to be made to achieve the organisational strategy. This framework can be adopted by any organisation that intends to use technology to implement organisational strategy. For those organisations that have already implemented their strategy, the framework can assist in improving their organisational strategy.

(2) The connectedness in selecting socio-technical theory to underpin information systems studies

This journal article was influenced by the fact that postgraduate students are struggling to choose a socio-technical theory to underpin their studies in the field of Information Systems (IS). The search was carried out on ten different socio-technical theories used in IS studies in the recent decade. A set of empirical data was analytical, extracted from Google scholar database, using criteria that included IS fields and year of publication. The descriptions about the theories were provided in a tabular format. Socio-technical theories such as actor network theory, structuration theory and diffusion of innovation were covered in this article.

7.4.2 Methodical contribution

The use of the moments of translation from the perspective of actor network theory (ANT) and innovation decision process from the perspective of diffusion of innovation (DOI) in the study is methodological, a contribution to teaching, learning and research in the field of information systems (IS). Prior to this study, the researcher struggled to identify any study where the two theories were combined in application within IS studies. This renders this particular result unique from other research that has been conducted. Actor network theory emphasised the four stages necessary for theoretically understanding the social-technical factors which interrelate during the testing and evaluation of software in the organisation. The five stages of innovation decision process from the perspective of diffusion of innovation were applied to diffuse the innovation within the organisation, which ANT lacked.

Without the combination of ANT and DOI, it would be difficult to determine the outcome of the study, from the data collection stage to the interpretation of the research. The scope, as defined by the four stages of moments of translation and five stages of innovation decision process, was vitally significant in this study making a difference. The difference, mainly in testing and evaluating of software within the organisation, includes how factors and actors connect, associate, relate and manifest themselves during the testing and evaluation of software. The theorised factors can now be put into practice by organisations intending to improve their performance in this field.

7.4.3 Practical contribution

The other contribution of the study is practical in nature. This is mainly because the findings of the study are factors which organisations could relate to in terms of their existence. Organisations are still facing the challenge of ensuring that software projects are tested and evaluated successfully so as not to disrupt business. However, the organisations tend to invest in technology and place less emphasis on equipping software testers with software testing knowledge and skills. Hence the same challenges continue to repeat themselves over the years. The contribution is mainly on empirical evidence, giving confidence to employers and employees in adjusting and managing the processes and activities in the testing and evaluation of software in their various organisations.

It is practical for the employers and employees to understand the research and relate to the findings, more easily eradicating the challenges by using the decision support system framework generated by this study.

7.5 Recommendations

The research has investigated various challenges encountered in the testing and evaluation of software in organisations. Organisations must pay attention to such challenges to ensure that they are prevented as early as possible in the software testing and evaluation process. In so doing, organisations would be able to achieve goals and objectives by producing quality software.

7.5.1 Documentation

Documentation, plays an essential role in software testing and evaluation, as it provides software testers with crucial information about what needs to be tested and evaluated. Documents, such as business requirement specifications, functional requirement specifications and technical requirement specifications, provide software testers with both functional and non-functional requirements about the software that needs to be tested and evaluated. The software testers rely on these documents to extract test requirements, test scenarios and test cases. Without these documents, software testers would be unclear about what to test.

The software under test is verified against the requirements specification. Any mismatch between the software and documents provided becomes a defect. Therefore, software testers would log that defect against a particular software developer to fix it. Once the defect is fixed, the software testers re-test the defect. Thereafter, they would perform regression testing to ensure that whatever was fixed has not affected the functionality that was previously working.

In some organisations, there is a tendency of neglecting to document the business requirements, and yet software testers are still expected to test and evaluate the software. As a result, the verification of software under test becomes a challenge because software testers would rely on what they are being told by software developers or what the software under test does. The quality of software is easily compromised due to lack of documentation.

7.5.2 Standards and procedures

Software testing is governed by software testing standards and procedures. These standards, are agreed upon by the international standard bodies such as ISO, to guide software testers in performing proper software testing. These standards and procedures enable organisations to benchmark themselves with the best practices in software testing. However, if these standards and procedures are bypassed quality software cannot be delivered. As software testing is process-oriented and procedural,

it is vital to follow the software testing standards and procedures. Failure to adhere to such standards leads to delivering poor quality software. If the organisation is affiliated with ISO, audit findings are expected from the registered organisation in order to verify that software was tested and evaluated accordingly. If the organisation still does not abide to the audit findings, they lose their affiliation. Therefore, the management needs to enforce these standards and procedures to produce quality software and retain their affiliations with ISO.

7.5.3 Quality of software

It is still not a guarantee that quality software would be produced when the organisation has an independent software testing team as well as sophisticated software testing tools. The organisation may have all this in place but fail to deliver as expected by the organisation. Here's why: it is vital to train the software testers to equip them with the necessary software testing skills. The software testers need to be passionate in what they are doing to take the organisation to the next level of quality. Management must support the needs of software testing team in terms of the good work they are doing. As organisations rely on quality software for competitiveness and sustainability, the management needs to enforce software testing standards and procedures to ensure that they are adhered to by all employees within the organisation. If the management fails in playing their part, employees are likely to ignore the software testing quality standards. As a result, the organisation may be out of competition due to poor quality software.

7.6 Benefit of the study

The benefits of the study are two-fold: first, it contributes to the body of knowledge and secondly, it undergirds to the organisation that deploys software. The benefits are discussed as follows:

The output of the study would contribute to the body of knowledge through literature. Many organisations, academic institutions and students depend on literature for their related work. The dedicated and in-depth nature of this study makes it authentic and gives others the confidence to apply it. Organisations can apply the decision support system framework and advance it regarding software testing and evaluation.

Moreover, the study illustrates that the testing and evaluation of software is of vital importance to organisations in this competitive environment. To a certain extent, many organisations are aware of some of the factors highlighted by the study, but too often ignored them. This was attributed to the fact that there was no empirical evidence for them. But it is crucial for organisations to be aware of such challenges to

ensure that they are prevented as early as possible in software testing and evaluation.

7.7 Further study

The study contributes to the body of knowledge from both a theoretical and practical perspective. Organisations invest so much money in developing software and enhancing existing software. However, most software is not utilised as it's supposed to be. This does not mean that the software is not used by many employees, but often they do not maximise its use as they are supposed to. Such software is referred to as 'white elephants'. According to Money Web, organisations such as the big four banks in South Africa spend around R30 billion on developing software each year. The reason why they spend so much money is because they continuously develop new software and enhance existing software. However, some of this software is not used and when new management is employed, often new software gets developed.

Software are not used because it is of low quality, a situation arising from a lack of proper software testing. If proper software testing and evaluation were performed, then quality software would be produced. As a result, employees would be happy to fully utilise such software. Having rigorously carried out this study, the researcher is confident in recommending that further research in the area of this study should be carried out. The factors influencing the testing and evaluation of software are needed to develop new software and enhance existing ones. The factors enabling the testing include documentation, relationships between team members, and heterogeneity of testers, standardisation, and procedural and implementation policies. Without this study, these factors would have not been established empirically. Also, the use of different theories such as structuration theory and activity theory for analysis could be applied for further studies.

7.8 Conclusion

This chapter has presented the conclusions drawn from the findings of this study, clearly establishing that the testing and evaluation of software has an enormous effect on IT projects. Failure to follow software testing standards and procedures can result in the software project team delivering poor quality software. All stakeholders involved in the software project team need to respect each other to work collaboratively in achieving quality software that would satisfy business requirements and customers. This study has been successful as it achieved all its objectives as articulated in Chapter 1 and repeated in Chapter 3 and Chapter 7. The empirical findings from the study will infuse confidence in management and sponsors for testing

and evaluating all software projects which are initiated for competitive advantage in organisations.

BIBLIOGRAPHY/REFERENCES

- Abbas, R., Sultan, Z. & Bhatti, S.N. 2017. Comparative analysis of automated load testing tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege. *In Communication Technologies (ComTech)*, 39-44.
- Abor, J. & Quartey, P. 2010. Issues in SME Development in Ghana and South Africa. *International Research Journal of Finance and Economics*, 39:218-228.
- Abu-Dalbouh, H.M. 2013. A Questionnaire Approach Based on the Technology Acceptance Model for Mobile Tracking on Patient Progress Applications. *Journal of Computer Science*, 9(6):763-770.
- Acharya, S. & Pandya, V. 2012. Bridge between Black Box and White Box – Grey Box. *International Journal of Electronics and Computer Science Engineering*, 2(1):175-185.
- Agarwal, S., Sharma, P. & Nikhil, K. 2012. A Review of the software testing process in SDLC. *International Journal of Electronics Communication and Computer Engineering*, 3(1):18-21.
- Ahamed, S.S.R. 2009. Study the feasibility and importance of software testing: An Analysis. *International Journal of Engineering Science and Technology*, 1(3):119-128.
- Aichernig, B.K. 2001. Systematic Black-Box Testing of Computer-Based Systems through Formal Abstraction Techniques. *Institute for Software Technology*.
- Alaqail, H. & Ahmed, S. 2018. Overview of Software Testing Standard ISO/IEC/IEEE 29119. *IJCSNS International Journal of Computer Science and Network Security*, 18(2):112-116.
- Alcouffe, S., Berland, N. & Levant, Y. 2008. Actor-networks and the diffusion of management accounting innovations: A comparative study. *Management Accounting Research*, 19:1-17.
- Alemneh, D.G. & Hastings, S.K. 2010. Exploration of Adoption of Preservation Metadata in Cultural Heritage Institutions: Case of PREMIS. *American Society for Information Science and Technology*.
- Alqahtani, S. & Wamba, S.F. 2012. Determinants of RFID Technology Adoption Intention in the Saudi Retail Industry: An Empirical Study. *45th Hawaii International Conference on System Sciences*, 4720-4729.
- Alsaawi, A. 2014. A Critical Review of Qualitative Interviews. *European Journal of Business and Social Sciences*, 3(4):149-156.
- Alshamrani, A. & Bahattab, A. 2015. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*, 12(1):106-111.
- Alshenqeeti, H. 2014. Interviewing as a Data Collection Method: A Critical Review. *English Linguistics Research*, 3(1):39-45.
- Amaratunga, D., Daldry, D., Sarshar, M. & Newton, R. 2002. Quantitative and qualitative research in the built environment: application of “mixed” research approach. *International Journal of Productivity and Performance Management*, 51(1):17-31.

- Amlani, R.A. 2012. Advantages and Limitations of Different SDLC Models. *International Journal of Computer Applications & Information Technology*, 1(3):6-11.
- Ansari, S., Panhwar, A.H. & Mahesar, G.A. 2016. Mixed Methods Research: Ontological, Epistemological and Methodological underpinnings. *International Research Journal of Language and Literature*, 27:133-141.
- Athanasiadis, A. & Andreopoulou, Z.S. 2011. DSS applications in forest policy and management: Analysis of current trends. *Proceeding of the International Conference on Information and Communication Technologies*, 549-557.
- Athanasiadis, A. & Andreopoulou, Z. 2015. A DSS for the identification of forest land types by the Greek Forest Service. *International Journal of Sustainable Agricultural Management and Informatics*, 1(1):76-88.
- Avison, D. & Fitzgerald, G. 2006. *Information Systems Development Methodologies, Techniques & Tools*. 4th ed. United Kingdom: McGraw-Hill.
- Azorin, J.M. & Cameron, R. 2010. The Application of Mixed Methods in Organisational Research: A Literature Review. *The Electronic Journal of Business Research Methods*, 8(2):95-105.
- Babbie, E. 2014. *The basics of social research*. 6th ed. Wadsworth, Cengage Learning, USA.
- Bamotra, A. & Randhawa, A.K. 2017. Software Testing Techniques. *International Journal of Innovative Computer Science & Engineering*, 4(3):122-126.
- Barker, J., Linsley, P. & Kane, R. 2016. *Evidence-based Practice for Nurses and Healthcare Professionals*. 3rd ed. Sage Publications Inc.
- Bassil, Y. 2012. A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering & Technology*, 2(5):742-749.
- Baxter, P. & Jack, S. 2008. Qualitative Case Study Methodology: Study Design and Implementation for Novice Researchers. *The Qualitative Report*, 13(4):544-559.
- Benbasat, I., Goldstein, D.K. & Mead, M. 1987. The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, 11(3):369-386.
- Besel, R.D. 2011. Opening the “Black Box” of Climate Change Science: Actor-Network Theory and Rhetorical Practice in Scientific Controversies. *Southern Communication Journal*, 76(2):120-136.
- Bhardwaj, S. 2015. Performance Testing Tools: A Comparative Analysis. *International Journal of Engineering Technology Management and Applied Sciences*, 3(4):100-105.
- Bhasin, A. & Kumar, M. 2015. Study of White Box, Black Box and Grey Box Testing Techniques. *International Journal of Research in Engineering & Advanced Technology*, 3(3):23-27.
- Bhattacharjee, A. 2012. *Social Science Research: Principles, Methods, and Practices*. 2nd ed. Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

- Bhatti, S. & Kumari, R. 2015. Comparative Study of Load Testing Tools. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(3):2334-2338.
- Biedenbach, T. & Jacobsson, M. 2016. The Open Secret of Values: The Roles of Values and Axiology in Project Research. *Project Management Journal*, 47(3):139-155.
- Binckman, L. & Rog, D.J. 2009. *The SAGE Handbook of Applied Social Research Methods*. 3rd ed. SAGE Publications, Inc.
- Bowleg, L. 2017. Towards a Critical Health Equity Research Stance: Why Epistemology and Methodology Matter More Than Qualitative Methods. *Health Education & Behavior*, 44(5):677–684.
- Broer, T., Nieboer, A.P. & Bal, R.A. 2010. Opening the black box of quality improvement collaboratives: An Actor-Network theory approach. *BMC Health Services Research*, 10(265):1-9.
- Brown, S.F. 2009. Naivety in systems engineering research: are we putting the methodological cart before the philosophical horse. *In 7th Annual Conference on Systems Engineering Research (CSER 2009)*.
- Bryman, A. 2008. *Social research methods*. 4th ed. Oxford University Press.
- Bukhari, A., Faisal, S. & Hira, K. 2014. A comparative study on usage of traditional and agile software development methodologies in software industry of Asia. *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*.
- Busse, J., Humm, B.G., Lübbert, C., Moelter, F., Reibold, A., Rewald, M., Schlüter, V., Seiler, B. & Tegtmeyer, E. 2015. Actually, What Does “Ontology” Mean? A Term Coined by Philosophy in the Light of Different Scientific Disciplines. *Journal of Computing and Information Technology*, 23(1):29–41.
- Bruun, H. & Hukkinen, J. 2003. Crossing Boundaries: An Integrative Framework for Studying Technological Change. *Social Studies of Science*, 33(1):5-116.
- Cameron, R. 2011. Mixed Methods Research: The Five Ps Framework. *Journal of Business Research Methods*, 9(2):96-108.
- Capaldo, G. & Rippa, P. 2009. A planned-oriented approach for EPR implementation strategy selection. *Journal of Enterprise Information Management*, 22(6):642-659.
- Chang, H.C. 2010. A New Perspective on Twitter Hashtag Use: Diffusion of Innovation Theory. *Proceedings of the American Society for Information Science and Technology*, 47(1):1-4.
- Chen, W., Zhang, C., Zheng, Y. & Ciu, L. 2009. The Interpretive Flexibility of an E-government Project: From an Actor-Network Theory Perspective. *Proceedings of the 42nd Hawaii International Conference on System Sciences, Hawaii*, 1-10.
- Cho, S., Mathiassen, L. & Nilsson, A. 2008. Contextual dynamics during health information systems implementation: an event-based actor-network approach. *European Journal of Information Systems*, 17(1):614–630.
- Cho, J.Y. & Lee, E.H. 2014. Reducing Confusion about Grounded Theory and Qualitative Content Analysis: Similarities and Differences. *The Qualitative Report*, 19(64):1-20.

- Cohen, L., Manion, L. & Morrison, K. 2007. *Research methods in education*. 6th ed. London: Routledge.
- Comber, A., Fisher, P. & Wadsworth, R. 2003. Actor–network theory: a suitable framework to understand how land cover mapping projects develop? *Land Use Policy*, 20(1):299–309.
- Cresswell, J.W. 2007. *Philosophical, Paradigm, and Interpretive Frameworks*. Qualitative inquiry & research design: choosing among five approaches. 2nd ed. Sage Publishers.
- Cresswell, K.M., Worth, A. & Sheikh, A. 2010. Actor-Network Theory and its role in understanding the implementation of information technology developments in healthcare. *BMC Medical Informatics and Decision Making*, 1-11.
- Creswell, J.W. 2009. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 3rd ed. Sage Publications, Thousand Oaks CA.
- Crossan, F. 2016. Research philosophy: towards an understanding. *Nurse Research*, 11(1):46-55.
- De Lisle, J. 2011. The Benefits and Challenges of Mixing Methods and Methodologies: Lessons Learnt from Implementing Qualitatively Led Mixed Methods Research Designs in Trinidad and Tobago. *Caribbean Curriculum*, 18:87-120.
- Department of Trade and Industry. 2010. THE COMPANIES ACT, NO. 71 OF 2008 AN EXPLANATORY GUIDE. Replacing the Companies Act, No. 61 of 1973. Available at: http://c.yimcdn.com/sites/www.iodsa.co.za/resource/collection/CB7E5DC1-E790-4BED-9693-9F8AA33E0032/Companies_Act_Guide.pdf (Accessed: 26 April 2018).
- Denney, A.S. & Tewksbury, R. 2012. How to Write a Literature Review. *Journal of Criminal Justice Education*, 1-17.
- De Wet, B. & Visser, J.K. 2013. An evaluation of software project risk management in SOUTH AFRICA. *South African Journal of Industrial Engineering*, 24(1):14-28.
- Dhiman, S. & Sharma, P. 2016. Performance Testing: A Comparative Study and Analysis of Web Service Testing Tools. *International Journal of Computer Science and Mobile Computing*, 5(6):507-512.
- DiCicco-Bloom, B. & Crabtree, B.F. 2006. The qualitative research interview. *Medical Education*, 40:314–321.
- Driscoll, D.L. 2011. Introduction to Primary Research: Observations, Surveys, and Interviews. *Writing Spaces: Readings on Writing*, 2:153-174.
- Dube, L. & Pare, G. 2003. Rigor in Information Systems Positivist Case Research: Current Practices, Trends, and Recommendations. *MIS Quarterly*, 27(4), 597-635.
- Dwivedi, Y.K., Henriksen, H.Z., Wastell, D. & De, R. 2013. *Grand Successes and Failures in IT*. IFIP WG 8.6 International Working Conference on Transfer and Diffusion of IT, TDIT 2013, Bangalore, India.

- Effah, J. 2012. Mobilizing Culture for E-Business in Developing Countries: An Actor Network Theory Account. *The Electronic Journal on Information Systems in Developing Countries*, 52(5):1-18.
- Ekdale, B., Singer, J., Tully, M. & Harmsen, S. 2015. Making Change: Diffusion of Technological, Relational, and Cultural Innovation in the Newsroom. *Journalism & Mass Communication Quarterly*, 92(4):938-958.
- Elo, S. & Kynga, S.H. 2008. The qualitative content analysis process. *Journal of Advanced Nursing*, 62(1):107–115.
- Engel, B.A., Choi, J.Y., Harbor, J. & Pandey, S. 2003. Web-based DSS for hydrologic impact evaluation of small watershed land use changes. *Computers and Electronics in Agriculture*, 39:241-249.
- Farooq, S.K. & Quadri, S. M. K. 2013. Empirical Evaluation of Software Testing Techniques - Need, Issues and Mitigation. *Software Engineering: An International Journal (SEIJ)*, 3(1):41-51.
- Feng, W. & Hannafin, M. J. 2005. Design-Based Research and Technology-Enhanced Environments. *Educational Technology Research and Development*, 53(4):5-23.
- Filip, F.G., Zamfirescu, C.B. & Ciurea, C. 2017. *Computer-Supported Collaborative Decision-Making*. Springer International Publishing.
- Fouka, G. & Mantzorou, M. 2011. What are the Major Ethical Issues in Conducting Research? Is there a Conflict between the Research Ethics and the Nature of Nursing? *Health Science Journal*, 5(1):3-14.
- Fuller, M.A., Hardin, A.M. & Scott, C.L. 2007. Diffusion of Virtual Innovation. *The DATA BASE for Advances in Information Systems*, 38(4):40-44.
- Gautam, S. & Nagpal, B. 2016. Descriptive Study of Software Testing & Testing Tools. *International Journal of Innovative Research in Computer and Communication Engineering*, 4(6):10288-10295.
- Gavric, G., Sormaz, G. & Ilic, D. 2016. The Impact of Organizational Culture on the Ultimate Performance of a Company. *Faculty of Business Economics and Entrepreneurship*, 3(4):25-30.
- Gelo, O., Braakmann, D. & Benetka, G. 2008. Quantitative and Qualitative Research: Beyond the Debate. *Integrative Psychological and Behavioral Science*, 42:266–290.
- Ghobakhloo, M., Sabouri, M.S., Hong, T.S. & Zulkifli, N. 2011. Information Technology Adoption in Small and Medium-sized Enterprises; An Appraisal of Two Decades Literature. *Interdisciplinary Journal of Research in Business*, 1(7):53-80.
- Ghobakhloo, M., Hong, T.S., Sabouri, M.S., & Zulkifli, N. 2012. Strategies for Successful Information Technology Adoption in Small and Medium-sized Enterprises. *Information*, 3:36-67.
- Gicheru, E. 2013. The psychology of unmarried men in Nairobi: A case study of three bachelors over forty. *African Journal of History and Culture*, 5(6):126-137.

- Goyette, S., Cassivi, L., Courchesne, M. & Elia, E. 2015. The ERP post-implementation stage: a knowledge transfer challenge. *International Journal of Information Systems and Project Management*, 3(2):5-19.
- Greenhalgh, T., Potts, H.W.W., Wong, G., Bark, P., & Swinglehurst, D. 2009. Tensions and Paradoxes in Electronic Patient Record Research: A Systematic Literature Review Using the Meta-narrative Method. *The Milbank Quarterly*, 87(4):729-788.
- Gunawong, P. & Gao, P. 2010. Challenges of eGovernment in Developing Countries: Actor-Network Analysis of Thailand's Smart ID Card Project. *Proceeding ICTD '10 Proceedings of the 4th ACM/IEEE International Conference on Information and Communication Technologies and Development*.
- Hanseth, O., Aanestad, M. & Berg, M. 2004. Guest editors' introduction Actor-network theory and information systems. What's so special? *Information Technology & People*, 17(2):116-123.
- Harris, L.R. & Brown, G.T. 2010. Mixing interview and questionnaire methods: Practical problems in aligning data. *Practical Assessment, Research & Evaluation*, 15(1):1-19.
- Hayes, B., Bonner, A. & Douglas, C. 2013. An introduction to mixed methods research for nephrology nurses. *Renal Society of Australasia Journal*, 9(1):8-14.
- Henderson, S. 2016. Research Methodology. *International Journal of Sales, Retailing and Marketing*, 4(9):1-97.
- Hernandez, R.A. & Fisher, B. 2013. A Qualitative Methodology for the Design of Visual Analytic Tools for Emergency Operation Centers. 46th Hawaii International Conference on System Sciences, 126-135.
- Hertz, P., Cavalieri, S., Finke, G.R., Duchi, A. & Schönsleben, P., 2013. A simulation-based decision support system for industrial field service network planning. *Simulation: Transactions of the Society for Modeling and Simulation International*, 1-16.
- Hertz, P., Cavalieri, S., Finke, G.R., Duchi, A. & Schönsleben, P., 2014. A simulation-based decision support system for industrial field service network planning. *Simulation*, 90(1):69-84.
- Hertzum, M., Bansler, J.P., Havn, E.C. & Simonsen, J. 2012. Pilot Implementation: Learning from Field Tests in IS Development. *Communications of the Association for Information Systems*, 30(1):313-328.
- Hilburn, T.B., Towhidnejad, M., Nangia, S. & Shen, L. 2006. A Case Study Project for Software Engineering Education. *36th ASEE/IEEE Frontiers in Education Conference. San Diego, CA*.
- Hoffman, D. 1999. Test Automation Architectures: Planning for Test Automation. *Software Quality Methods*.
- Hooda, I. & Chhillar, R.S. 2015. Software Test Process, Testing Types and Techniques. *International Journal of Computer Applications*, 111(13):10-14.
- Hosio, S., Goncalves, J., Anagnostopoulos, T. & Kostakos, V. 2016. Leveraging wisdom of the crowd for decision support. *In Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion!*

- Houghton, C., Hunter, A. & Meskell, P. 2012. Linking aims, paradigm and method in nursing research. *Nurse Researcher*, 20(2):34-39.
- Hussain, T. & Singh, S. 2015. A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing. *IJAPRR International Journal Peer Reviewed Refereed*, 2(5):1-8.
- Ihuah, P.W. & Eaton, D. 2013. The Pragmatic Research Approach: A Framework for Sustainable Management of Public Housing Estates in Nigeria. *Journal of US-China Public Administration*, 10(10):933-944.
- Irena, J. 2008. Software Testing Methods and Techniques. *The IPSI BgD Transactions on Internet Research*, 30-41.
- ISO/IEC/IEEE International Standard. 2013. Software and systems engineering -- Software testing --Part 3: Test documentation. ISO/IEC/IEEE 29119-1:2013(E), 1-64. IEEE.
- Iyamu, T. & Roode, D. 2010. The Use of Structuration Theory and actor Network Theory for analysis: Case Study of a financial Institution in South Africa. *International Journal of Actor-Network Theory and Technological Innovation*, 2(1):1-26.
- Iyamu, T. 2011. Institutionalisation of the Enterprise Architecture: The Actor-Network Perspective. *International Journal of Actor-Network Theory and Technological Innovation*, 3(1):27-38.
- Iyamu, T. 2013. Underpinning theories: order-of-use in information systems research. *Journal of Systems and Information Technology*, 15(3), 224-238.
- Iyamu, T. & Sekgweleo, T. 2013. Information Systems and Actor-Network Theory Analysis. *International Journal of Actor-Network Theory and Technological Innovation*, 5(3):1-11.
- Iyamu, T., Sekgweleo, T. & Mkhomazi, S.S. 2013. Actor Network Theory in Interpretative Research Approach. *IFIP International Federation for Information Processing*, 605-610.
- Jain, R. & Raju, S.S. 2016. *Decision Support System in Agriculture using Quantitative Analysis*. Agrotech Publishing Academy.
- Jamil, M.A., Arif, M., Abubakar, N.S.A. & Ahmad, A. 2016. Software Testing Techniques: A Literature Review. *6th International Conference on Information and Communication Technology for The Muslim World*, 177-182.
- Jamshed, S. 2014. Qualitative research method-interviewing and observation. *Journal of Basic and Clinical Pharmacy*, 5(4):87-88.
- Jan, S.R., Shah, S.T.U., Johar, Z.U., Shah, Y. & Khan, F. 2016. An Innovative Approach to Investigate Various Software Testing Techniques and Strategies. *International Journal of Scientific Research in Science, Engineering and Technology*, 2(2):682-689.
- Kannan, V., Jhajharia, S. & Verma, S. 2014. Agile vs waterfall: A Comparative Analysis. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 3(10):2680-2686.
- Kapur, P.K., Yadavalli, V.S. & Kumar, A. 2006. A General Software Reliability Growth Model for a Distributed Environment. *South African Statistical Journal*, 151-185.

- Kasurinen, J. 2012. Software Organizations and Test Process Development. *Advances in Computers*, 85:1-63.
- Kaur, M. & Kumari, R. 2011. Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro. *International Journal of Computer Applications*, 24(1):1-7.
- Khan, M.E. 2011. Different Approaches to White Box Testing Technique for Finding Errors. *International Journal of Software Engineering and Its Applications*, 5(3):1-14.
- Khan, M.E. & Khan, F. 2012. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *International Journal of Advanced Computer Science and Applications*, 3(6):12-15.
- Khojasteh, J., Zeki, T.S., Naji, H. R. & Sanatnama, H. 2012. A Functional Testing Modeling for Enhanced Software Testing. *International Journal of Science and Technology*, 2(10):738-741.
- Kivunja, C. & Kuyini, A.B. 2017. Understanding and Applying Research Paradigms in Educational Contexts. *International Journal of Higher Education*, 6(5):26-41.
- Kolhe, P.R., Khetri, G.P. & Deshmukh, N.K. 2013. Study of Standard Assumptions of Graphical User Interface (GUI) Based on Usability, Adaptability and Security Factors. *International Journal of Emerging Research in Management & Technology*, 2(12):92-95.
- Kothari, C.R. 2004. *Research Methodology Methods and Techniques*. 2nd ed. New Age International Publishers.
- Koza, N. 2015. *FLY SAFAIR'S R1 BARGAIN CAUSES WEBSITE TO CRASH*. [online] Available at: <http://ewn.co.za/2015/08/25/Fly-Safairs-R1-bargain-causes-website-crash>. [Accessed 26 May 2018].
- Lee, H., & Oh, S. 2006. A standards war waged by a developing country: Understanding international standard setting from the actor-network perspective. *Journal of Strategic Information Systems*, 15:177-195.
- Lee, M. 2014. Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance. *British Journal of Applied Science & Technology*, 4(21):3070-3095.
- Leeuw, E.D., Hox, J.J. & Dillman, D.A. 2008. *International Handbook of Survey Methodology*. European Association of Methodology.
- Levers, M.J.D. 2013. Philosophical Paradigms, Grounded Theory, and Perspectives on Emergence. *Sage Publications Inc*, 1-6.
- Lihosit, J. 2014. Breaking Down the Black Box: How Actor Network Theory Can Help Librarians Better Train Law Students in Legal Research Techniques. *Law Library Journal*, 106(2):211-220.
- Liu, S., Duffy, A., Whitfield, R. I. & Boyle, I. M. 2010. Integration of decision support systems to improve decision support performance. *Knowledge and Information Systems*, 22(3):261-286.
- Lokke, A. & Sorensen, P. 2014. Theory Testing Using Case Studies. *The Electronic Journal of Business Research Methods*, 12(1):66-74.

- Loukis, E., Charalabidis, Y. & Androutsopoulou, A. 2017. Promoting open innovation in the public sector through social media monitoring. *Government Information Quarterly*, 34:99–109.
- Luo, Q. 2016. Input-sensitive performance testing. *In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 1085-1087.
- Luoma-Aho, V. & Paloviita, A. 2010. Actor-networking stakeholder theory for today's corporate communications. *Corporate Communications: An International Journal*, 15(1):49-67.
- Luqman, A., Abdullah, N.K. & Ghapar, M.A. 2011. Modeling the Adoption of E-business amongst SMEs in Terengganu. *IEEE Colloquium on Humanities, Science and Engineering Research (CHUSER 2011)*.
- Mack, L. 2010. The Philosophical Underpinnings of Educational Research. *Polyglossia*, 19:5-11.
- Mackenzie, N. & Knipe, S. 2006. Research dilemmas: Paradigms, methods and methodology. *Issues in Educational Research*, 16:193-205.
- Mahring, M., Holmstrom, J., Keil, M. & Montealegre, R. 2004. Trojan actor-networks and swift translation Bringing actor-network theory to IT project escalation studies. *Information Technology & People*, 17(2):210-238.
- Mandi, G. & Kumar, P. 2013. Reducing the size of test suite using a variant of Non-dominated Sorting Genetic Algorithm II. *International Journal of Advance Research in Computer Science and Management Studies*, 1(6):66-75.
- Matalonga, S., Rodrigues, F. & Travassos, G.H. 2015. Matching context aware software testing design techniques to ISO/IEC/IEEE 29119. *In International Conference on Software Process Improvement and Capability Determination*, 33-44.
- Merriam, B.M. & Tisdell, E.J. 2015. *Qualitative Research: A Guide to Design and Implementation*. 4th ed. John Wiley and Sons.
- Mishra, D., Ostrovska, S. & Hacaloglu, T. 2017. Exploring and expanding students' success in software testing. *Information Technology & People*, 30(4): 927-945.
- Mishra, S. & Pradhan, A. 2012. Software Testing Techniques Adopted in Corporate Sectors: A Precise Study. *International Journal of Emerging Technology and Advanced Engineering*, 2(9):290-295.
- Monama, T., Ndlazi, S. & Mabotja, K. 2016. *Online applications: Bring it on, says Lesufi*. [online] Available at: <https://www.iol.co.za/lifestyle/family/parenting/online-applications-bring-it-on-says-lesufi-2011891>. [Accessed 26 May 2018].
- Monier, M. & El-mahdy, M.M. 2015. Evaluation of automated web testing tools. *International Journal of Computer Applications Technology and Research*, 4(5):405-408.
- Montfort, D., Brown, S. & Pegg, J.M. 2009. An Investigation of the Adoption of an Assessment Instrument for Capstone Design Courses. *Proceedings of the 39th IEEE international conference on Frontiers in education conference*, 148-153.

Morgan, D.L. 2007. Paradigms Lost and Pragmatism Regained: Methodological Implications of Combining Qualitative and Quantitative Methods. *Journal of Mixed Methods Research*, 1(1):48-76.

Munassar, N.M. & Govardhan, A. 2010. A Comparison Between Five Models of Software Engineering. *IJCSI International Journal of Computer Science Issues*, 7(5):94-101.

Muscatello, J.R., Small, M.H. & Chen, I.J. 2003. Implementing enterprise resource planning (ERP) systems in small and midsize manufacturing firms. *International Journal of Operations & Production Management*, 23(8):850-871.

Nabukenya, J. 2012. Combining Case Study, Design Science and Action Research Methods for Effective Collaboration Engineering Research Efforts. *45th Hawaii International Conference on System Sciences*, 343-352.

Nahas, M. & Maaita, A. 2012. Choosing Appropriate Programming Language to Implement Software for Real-Time Resource-Constrained Embedded Systems. In *Embedded Systems-Theory and Design Methodology. InTech*.

National small business Act, 1996. [online] Available at: https://www.thedti.gov.za/sme_development/docs/act.pdf. [Accessed 09 April 2018].

Nawaz, A., & Malik, K.M. 2008. *Software Testing Process in Agile Development*. Department of Computer Science School of Engineering, Blekinge Institute of Technology

Nemutanzhela, P. & Iyamu, T. 2011. A Framework for Enhancing the Information Systems Innovation: Using Competitive Intelligence. *The Electronic Journal Information Systems Evaluation*, 14:242-253.

Nenty, H.J. 2009. Writing a Quantitative Research Thesis. *Int J Edu Sc*, 1(1):19-32.

Nidhra, S. & Dondeti, J. 2012. Black box and white box techniques - A literature review. *International Journal of Embedded Systems and Applications*, 2(2):29-50.

Oates, B.J. 2006. *Researching information systems and computing*. 1st ed. Thousand Oaks, California. Sage Publications, Inc.

Okrent, M.D. & Vokurka, R.J. 2004. Process mapping in successful ERP implementations. *Industrial Management & Data Systems*, 104(8):637-643.

Olsson, A., Skovdahl, K. & Engström, M. 2016. Using diffusion of innovation theory to describe perceptions of a passive positioning alarm among persons with mild dementia: a repeated interview study. *BMC geriatrics*, 16(1):1-6.

Oppong, S. 2014. A Critique of the Philosophical Underpinnings of Mainstream Social Science Research. *Academicus International Scientific Journal*, 10:242-254.

Overhage, S. & Schlauderer, S. 2012. Investigating the Long-Term Acceptance of Agile Methodologies: An Empirical Study of Developer Perceptions in Scrum Projects. *2012 45th Hawaii International Conference on System Sciences*, 5452-5461.

Petersen, K. & Gencel, C. 2013. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In *Software Measurement and the 2013*

- Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 81-89.
- Petrescu, M. & Lauer, B. 2017. Qualitative Marketing Research: The State of Journal Publications. *The Qualitative Report*, 22(9):2248-2287.
- Potts, L. 2009. Using Actor Network Theory to Trace and Improve Multimodal Communication Design. *Technical Communication Quarterly*, 18(3):281-301.
- Qazi, A.S., Shahzadi, S. & Humayun, M. 2016. A Comparative Study of Software Inspection Techniques for Quality Perspective. *International Journal of Modern Education and Computer Science*, 8(10):9-16.
- Quadri, S.M.K. & Farooq, S.U. 2010. Software Testing – Goals, Principles, and Limitations. *International Journal of Computer Applications*, 6(9):7-10.
- Raadschelders, J.C.N. 2011. The Future of the Study of Public Administration: Embedding Research Object and Methodology in Epistemology and Ontology. *Public Administration Review*, 916-924.
- Ragab, M.A.F. & Arisha, A. 2018. Research Methodology in Business: A Starter's Guide. *Management and Organizational Studies*, 5(1):1-14.
- Rajasekar, S., Philominathan, P. & Chinnathambi, V. 2013. Research Methodology. *Physics.gen-ph*, 1-53.
- Raus, M., Flügge, B. & Boutellier, R. 2008. Innovation Steps in the Diffusion of e-Customs Solutions. *The Proceedings of the 9th Annual International Digital Government Research Conference*, 316-324.
- Reiter, B. 2013. The Epistemology and Methodology of Exploratory Social Science Research: Crossing Popper with Marcuse. *Government and International Affairs Faculty Publications*, 1-17.
- Rhodes, J. 2009. Using Actor-Network Theory to Trace an ICT (Telecenter) Implementation Trajectory in an African Women's Micro-Enterprise Development Organization. *Information Technologies and International Development*, 5(3):1-20.
- Saleh, M.F. 2011. An Agile Software Development Framework. *International Journal of Software Engineering (IJSE)*, 2(5):97-106.
- Sampaio, L., Varajão, J., Pires, E.J.S. & de Moura Oliveira, P.B. 2012. Diffusion of Innovation in Organizations: Simulation using Evolutionary Computation. *2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 25-30.
- Sang, H.A. & Tsai, D.R. 2009. Analyzing Strategies of Integrating ICT into Teaching Activities Using Innovation Diffusion Theory. *Fifth International Joint Conference on INC, IMS and IDC*, 1876 -1878.
- Saravanan, K. & Prasad, P.C.P. 2016. Open Source Software Test Automation Tools: A Competitive Necessity. *International Journal of Management & Development*, 3(6):103-110.
- Saunders, M., Lewis, P. & Thornhill, A. 2007. *Research methods for business students*. 4th ed. London: Financial Times Prentice Hall.

- Saunders, M., Lewis, P. & Thornhill, A. 2009. *Research methods for business students*. 5th ed. Pearson Education Limited.
- Saunders, M., Lewis, P. & Thornhill, A. 2012. *Research Methods for Business Students*. 6th ed. Pearson Education Limited.
- Sawant, A.A., Bari, P.H. & Chawan, P.M. 2012. Software Testing Techniques and Strategies. *International Journal of Engineering Research and Applications*, 1(3):980-986.
- Saxena, R. & Singh, M. 2014. Grey Box Testing: Proactive Methodology for the Future Design of Test Cases to Reduce Overall System Cost. *Journal of Basic and Applied Engineering Research*, 1(8):62-66.
- Schlegel, D. 2015. *Cost-of-Capital in Managerial Finance, Contributions to Management*. Springer International Publishing Switzerland.
- Scott, E., Zadirov, A., Feinberg, S. & Jayakody, R. 2004. The Alignment of Software Testing Skills of IS Students with Industry Practices – A South African Perspective. *Journal of Information Technology Education*, 3:161-172.
- Scotland, J. 2012. Exploring the Philosophical Underpinnings of Research: Relating Ontology and Epistemology to the Methodology and Methods of the Scientific, Interpretive, and Critical Research Paradigms. *English Language Teaching*, 5(9):9-16.
- Sefotho, M.M. 2015. A Researcher's Dilemma: Philosophy in Crafting Dissertations and Theses. *Journal of Social Science*, 42(1,2):23-36.
- Sekgweleo, T. 2015a. Understanding Traditional Systems Development Methodologies. *International Journal of Advances in Management and Economics*, 4(3):51-58.
- Sekgweleo, T. 2015b. Understanding Agile System Development Methodologies. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(7):18-24.
- Sembiring, J. & Adi, S. 2015. Business Value of Information Technology Service Quality Based on Probabilistic Business-Driven Model. *Journal of ICT Research and Applications*, 9(1):39-67.
- Sener, Z. & Karsak, E.E. 2012. A decision model for setting target levels in software quality function deployment to respond to rapidly changing customer needs. *Concurrent Engineering Research and Applications*, 20(1):19-29.
- Sharmila, S. & Ramadevi, E. 2014. Analysis of Performance Testing on Web Applications. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(3):5258-5260.
- Shao, D., Khurshid, S. & Perry, D.E. 2007. A Case for White-box Testing Using Declarative Specifications Poster Abstract. *Testing: Academic and Industrial Conference Practice and Research Techniques*, 137-137.
- Shin, S.Y. 2014. Two epistemological paradigms of self-management intervention for older adults with osteoarthritis. *Japan Journal of Nursing Science*, 11:144–149
- Shrivastava, S.V. & Date, H. 2010. Distributed Agile Software Development: A Review. *Journal of Computer Science and Engineering*, 1(1):10-17.

- Singh, I. & Tarika, B. 2014. Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir. *International Journal of Information & Computation Technology*, 4(15):1507-1518.
- Singla, S. & Kaur, H. 2014. Selenium Keyword Driven Automation Testing Framework. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(6):125-129.
- Skidmore, S. 2006. The V-model. *Professional Scheme Paper*, 2:48-49.
- Sowunmi, O.Y., Misra, S., Fernandez-Sanz, L., Crawford, B. & Soto, R. 2016. An empirical evaluation of software quality assurance practices and challenges in a developing country: a comparison of Nigeria and Turkey. *SpringerPlus*, 5(1):1-13.
- Steegmans, E., Bekaert, P., Devos, F., Delanote, G., Smeets, N., Van Dooren, M. & Boydens, J. 2004. Black & White Testing: Bridging Black Box Testing and White Box Testing. *Conferentie Software Testing: Beheers Optimaal de Risico's van IT in uw Business*, 1-12.
- Steen, J. 2010. Actor-network theory and the dilemma of the resource concept in strategic management. *Scandinavian Journal of Management*, 26(3):324-331.
- Sutton, J. & Austin, Z. 2015. Qualitative Research: Data Collection, Analysis, and Management. *The Canadian Journal of Hospital Pharmacy*, 68(3): 226–231.
- Soroka, V. & Jacovi, M. 2004. The Diffusion of ReachOut: Analysis and Framework for the Successful Diffusion of Collaboration Technologies. *Proceeding CSCW '04 Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 6(3):314-323.
- Sweis, R.J., Isa, A., Azzeh, H., Shtyh, B., Musa, E. & Albtoush, R.M. 2014. Nurses' Resistance to the Adoption of Information Technology in Jordanian Hospitals. *Life Science Journal*, 11(4):8-18.
- Tarrant, H. 2016. *Big Four' IT spending tops R30 billion a year*. [online] Available at: <https://www.moneyweb.co.za/news/tech/big-four-it-spending-tops-r30-billion-a-year>. [Accessed 26 May 2018].
- Tatnall, A. & Gilding, A. 1999. Actor-Network Theory and Information Systems. *Proc. 10th Australasian Conference on Information Systems*, 955-966.
- Tatnall, A. 2014. *Technological Advancements and the Impact of Actor-Network Theory*. Information Science Reference IGI Global.
- Teles, A. & Joia, L.A. 2011. Assessment of digital inclusion via the actor-network theory: The case of the Brazilian municipality of Pirai. *Telematics and Informatics*, 28(3):191-203.
- Tennis, J.T. 2008. Epistemology, Theory, and Methodology in Knowledge Organization: Toward a Classification, Metatheory, and Research Framework. *In Knowledge Organization*, 35(2/3):102-112.
- Terreberry, S.C. 2017. Understanding Student and Faculty Perceptions of the Accommodation and Support Procedures for Students with LD in Ontario Universities: A Mixed Methods Approach. *Electronic Thesis and Dissertation Repository*.
- Thanh, N.C. & Thanh, T.T. 2015. The Interconnection between Interpretivist Paradigm and Qualitative Methods in Education. *American Journal of Educational Science*, 1(2):24-27.

- Timpka, T., Bang, M., Delbanco, T. & Walker, J. 2007. Information infrastructure for inter-organizational mental health services: An actor network theory analysis of psychiatric rehabilitation. *Journal of Biomedical Informatics*, 40:429-437.
- Tretmans, J. 1999. Testing Concurrent Systems: A Formal Approach. *International Conference on Concurrency Theory*, 46-66.
- Tripathi, K.P. 2011. Decision support system is a tool for making better decisions in the organisation. *Indian Journal of Computer Science and Engineering (IJCSSE)*, 2(1):112-117.
- Tumuhairwe, K.G. 2013. Analysis of Library and Information Science/Studies (LIS) Education Today: The Inclusion of Indigenous Knowledge and Multicultural Issues in LIS Curriculum. *Creative Commons Attribution 3.0 Unported License*, 1-20.
- Van Der Duim, R. 2007. Tourismscapes an Actor-Network Perspective. *Annals of Tourism Research*, 34(4):961-976.
- Van Dijk, R.W. 2011. Determining the Suitability of Agile Methods for a Software Project. *University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science*.
- Van Iddekinge, C.H., Raymark, P.H. & Roth, P.L. 2006. Comparing the psychometric characteristics of ratings of face-to-face and videotaped structured interviews. *International Journal of Selection and Assessment*, 14(4):347-359.
- Vinekar, V., Slinkman, C.W. & Nerur, S. 2006. Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management*, 31-42.
- Waje, S., Gaikwad, V. & Chaudhari, P. 2014. Software Testing, Mythology & Methodologies. *International Journal of Emerging Technology and Advanced Engineering*, 4(2):673-677.
- Wedawatta, G., Ingririge, B. & Amaratunga, D. 2011. Case study as a research strategy: Investigating extreme weather resilience of construction SMEs in the UK. *7th Annual International Conference of International Institute for Infrastructure, Renewal and Reconstruction*.
- Whiting, L.S. 2008. Semi-structured interviews: guidance for novice researchers. *Nursing standard*, 22(23):35-40.
- Wijesinghe, G. 2009. Philosophy and methodological tradition of hermeneutics and phenomenology in researching' lived experience. *3rd Critical Tourism Studies Conference Proceedings*, 161-171.
- Williams, L. 2006. White-Box Testing. [online] Available at: <https://students.cs.byu.edu/~cs340ta/spring2018/readings/WhiteBox.pdf>. [Accessed 05 February 2018].
- Williams-Jones, B. & Graham, J.E. 2003. Actor-Network Theory: a toll to support ethical analysis of commercial genetic testing. *New Genetics and Society*, 22(3):271-296.
- Willis, J.W. 2007. *Foundations of qualitative research: interpretive and critical approaches*. London: Sage.
- Yan, K. 2009. Research into Behaviors at Initial Stage of IT and Courses Integration Project Based on Innovative Diffusion. *Intelligent Systems and Applications*, 1-4.

- Yang, Z., Wang, X. & Su, C. 2006. A review of research methodologies in international business. *International Business Review*, 15(6):601-617.
- Yanow, D. & Schwartz-Shea, P. 2011. *Interpretive Approaches to Research Design: Concepts and Processes*. Netherlands: Routledge.
- Yazan, B. 2015. Three Approaches to Case Study Methods in Education: Yin, Merriam, and Stake. *The Qualitative Report*, 20(2):134-152.
- Yilmaz, K. 2013. Comparison of Quantitative and Qualitative Research Traditions: epistemological, theoretical, and methodological differences. *European Journal of Education*, 48(2):311–325.
- Yin, R.K. 2013. Validity and generalization in future case study evaluations. *Evaluation*, 19(3):321-332.
- Yin, R.K. 2014. *Case Study Research Design and Methods*. 5th ed. Thousand Oaks, CA: Sage.
- Zalaghi, H. & Khazaei, M. 2016. The Role of Deductive and Inductive Reasoning in Accounting Research and Standard Setting. *Asian Journal of Finance & Accounting*, 8(1):24-37.
- Zhai, C. 2011. B2B e-marketplace adoption in China: from the perspective of innovation diffusion theory and network externalities. *IEEE*.
- Zolkepli, I.A. & Kamarulzaman, Y. 2015. Social media adoption: The role of media needs and innovation characteristics. *Computers in Human Behavior*, 43:189-209.
- Zuber-Skerritt, O. & Fletcher, M. 2007. The quality of an action research thesis in the social sciences. *Quality Assurance in Education*, 15(4):413-436.

APPENDICES

APPENDIX A: Interview Questions

Interview Questions

1. How would you describe the way in which software is tested and evaluated within your organisation?
2. What are some of the processes involved in testing and evaluating software?
3. In your view, what do you consider significant factors in testing the software?
4. What are some of the factors considered during software testing and evaluation?
5. What do you think are the implications of those factors?
6. What do you think is the role of technology such as software and hardware in the testing and evaluation of software?
7. What do you think are the challenges encountered during software testing and evaluation?
8. Why do you think those challenges are there?
9. What are the roles of people during the testing and evaluation of software?
10. What are some of the perceptions of employees with regard to software testing?
11. Could you please share some of your experiences in the testing and evaluation of software within your organisation?

APPENDIX B: Ethical Consideration Letter



P.O. Box 652 • Cape Town 8000 South Africa • Tel: +27 21 469 1012 • Fax +27 21 469 1002
80 Roeland Street, Vredehoek, Cape Town 8001

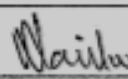
Office of the Research Ethics Committee	Faculty of Informatics and Design
--	-----------------------------------

Approval was granted by the Faculty Research Ethics Committee, on 1 September 2016 to Mr Tefo Gordon Sekgweleo, student number 216247500, for research activities related to the DTech: Information Technology degree at the Faculty of Informatics and Design, Cape Peninsula University of Technology.

Title of dissertation/thesis:	A decision support system framework for testing and evaluating software in organisations
-------------------------------	--

Comments

Research activities are restricted to those detailed in the research proposal.

 Signed: Faculty Research Ethics Committee	<u>1 Sept. 2016.</u> Date
--	------------------------------