Knowledge Management in Learning Software SMMEs in KwaZulu-Natal, South Africa

Mzwandile Muzi Shongwe
Student No. 211550432

Submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Information Studies, at the School of Social Sciences, College of Humanities, University of KwaZulu-Natal, Pietermaritzburg, South Africa

Supervisor: Professor Stephen Mutula,
University of KwaZulu-Natal

December 2017

DECLARATION

Submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in the Graduate Programme in Information Studies, School of Social Sciences, College of Humanities, University of KwaZulu-Natal, Pietermaritzburg, South Africa.

I, **Mzwandile Muzi Shongwe**, declare that:

1. The research reported in this thesis, except where otherwise indicated, is my original research.

2. This thesis has not been submitted for any degree or examination at any other university.

3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

5. Their words have been re-written but the general information attributed to them has been referenced

6. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.

7. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References section.

_____

Student Name

_____

Date

_____

Name of Supervisor

_____

Signature

Abstract

The study investigated the nature and causes of software development failures and knowledge management practices adopted to mitigate the failures in small, micro, and medium software developing enterprises (SMMEs) in the province of KwaZulu-Natal, South Africa.

The study adopted an interpretive, qualitative multiple case study approach to investigate the problem. Twelve software development SMMEs were involved in the study. Interviews were conducted with 12 information technology (IT)/software development project managers and eight software developers identified through purposive sampling. Qualitative content analysis was used to analyse and interpret the data.

The findings reveal that software development SMMEs in the province of KwaZulu-Natal, South Africa, experience software development failures. Ten causes of failure were identified. They are bureaucracy in IT departments, compatibility issues, complacency of developers, involvement of the wrong people in the planning stages of projects, a lack of detailed documentation, lack of resources, lack of user commitment/non-adoption of systems, miscommunication/misrepresentation of requirements, unrealistic customer expectations, and work overload. The results also indicate that software organisations and individual software developers experience knowledge gaps during the course of their work. Six knowledge management practices are adopted by the organisations and the individual developers to fill the knowledge gaps. The practices are knowledge acquisition, creation, storage, sharing, organisation and application. These practices are supported by Internet technologies such as blogs, Wikis, search engines, social networks, organisational databases and computer hardware such as servers and personal computers.

The study reveals two important knowledge management practices that are ignored by software organisations, namely post-mortem reviews, which are essential in software development, and formal training of the developers. The findings further reveal that knowledge management has enabled the organisations and individual developers to save time, retain their intellectual property (IP), become more efficient and effective in knowledge reuse. Organisations face a number of knowledge management related challenges. The challenges are lack of formal knowledge management procedures, difficulty protecting knowledge, expensive knowledge storage costs, increasing information needs, lack of the

time to fully adopt knowledge management practices, difficulty finding information, and the ever-changing nature of knowledge.

The study concluded that software development failures are prevalent in software SMMEs and that the organisations have informally adopted knowledge management. Moreover, knowledge management has brought benefits to the organisations but the role played by knowledge management in eliminating project failures is not clear.

It is recommended that software organisations should consider formally adopting knowledge management so that knowledge management specialists can be employed to drive the knowledge management initiatives and so help in conducting post-mortem reviews and the training of staff. In addition, further research is recommended to investigate the role of knowledge management in reducing or eliminating software project failures. Quantitative studies are also recommended to objectively measure the benefits brought by knowledge management. Such studies would measure how much time and which costs are saved by adopting knowledge management.

The study contributes to theory and practice (software development industry). Theoretically, the study developed and used a conceptual framework developed from software engineering and knowledge management that could be used to investigate knowledge management activities in organisations. The study also contributes to the existing body of knowledge on the subject software learning organisations from a developing country perspective. It is envisaged that software development organisations will adopt the recommendations proffered to improve their knowledge management practices.

Acknowledgments

I would like to extend my sincere thanks to the people mentioned below. If it was not for you, this work could not have been completed.

Firstly, I would like to thank the Universities of Zululand and KwaZulu-Natal for funding this study. I would like to acknowledge my supervisor, Professor S.M. Mutula for his guidance in the preparation of this thesis. To my mother, Sibongile C. Dlamini, thank you for everything. I would also like to extend my gratitude to the Department of Information Studies, University of Zululand, especially Professor B.J. Mostert, for giving me time off to collect data and for her encouragement during my study for the PhD. To Professor Dennis Ocholla, thank you "Mzee" for the mentorship and scholarly support. To my aunt, Mrs Brenda Mhlanga, you were always concerned about my studies. Thank you so much for your trust in my academic abilities.

To the following managers who sacrificed their time to respond to my questions:

- Alan Hastings and your team
- Anthony Tony' de Wijn
- Bernd von Fintel
- Bob Ludlow and your team
- Francois Nortie and your team
- Howard Hudson
- John Kiln and your team
- Rakesh Lutchman and your team
- Ryan Pretorius
- Salaelo Tlabela 'Sly' and your team
- Sue Ramgulam and Bruce
- Terry Duff and your team

Thank you so much for your contributions.

To everyone who wished me well, thank you!

Dedication

This work is dedicated to my mother, Sibongile Claudia Dlamini, who through thick and thin stood by me. We have fought, lost and conquered many battles together, Mom. If it was not for your love, support and confidence in me, Mlangeni, probably I would be on the streets today.

This is for you and your grandchildren!

Table of contents

Contents

List of acronyms

BRICS:	Brazil, Russia, India, China, South Africa
CASE:	Computer aided software engineering
CMM:	Capability maturity model
CMMI:	Capability maturity model integration
ERP:	Enterprise resource planning
FTP:	File transfer protocol
GDP:	Gross domestic product
GSD:	Global software development
GIS:	Geographic information system
HTML:	Hypertext Markup Language
IBM:	International Business Machines
IOSAEC:	Identify, organise, store, apply, evaluate, and create
IP:	Intellectual property
IS:	Information systems
ISO:	International Standardisation Organisation
IT:	Information technology
JAD:	Joint application development
JCSE:	Johannesburg Centre for Software Engineering
KZN:	KwaZulu-Natal
NASA:	National Aeronautics and Space Administration
OECD:	Organisation for Economic Cooperation and Development
OSD:	Open-source development
PDF:	Portable document format
RAD:	Rapid application development

REP:	Requirements elicitation process
RUP:	Rational unified process
SA:	South Africa
SAP:	System Analysis and Program Development
SDLC:	Systems development lifecycle
SECI:	Socialisation, externalisation, combination, internalisation
SEDA:	Small Enterprise Development Agency
SIDLC:	Systems integration lifecycle
SMME:	Small, micro, and medium enterprise
SMS:	Short messaging service
SPI:	Software process improvement
SPICE:	Software Process Improvement and Capability determination
SQL:	Structured query language
UKZN:	University of KwaZulu-Natal
USA:	United States of America
XP:	Extreme programming

List of figures

List of tables

Appendices

CHAPTER ONE
INTRODUCTION

1.1. Background to the research problem

Many challenges face software development organisations, globally and in South Africa. The challenges include among others, lack of software development skills, poor software quality, projects not being completed on time, and software projects exceeding their budget, (Dey, Kinch & Ogunlana, 2007; Charette, 2010; Tarawneh, 2011; Gruner & van Zyl, 2011). The Standish Group CHAOS report (2015) and the ITWeb survey (2013) indicate that less than 50% of projects are successful. There are many reasons why software projects fail. This issue has been of great concern since the start of software development in the 1970s. Among the reasons why software development projects fail are unrealistic or unarticulated project goals, lack of technical skills, lack of motivation, the collapse in organisational intelligence and failure to learn from past mistakes (Dey *et al*., 2007; Cerpa & Verner, 2009; Lehtinen *et al*., 2014). Gruner and van Zyl (2011) assert that South African software organisations experience similar problems.

Many frameworks and models have been designed in the software development industry to address software project failures. Software process improvement (SPI) frameworks such as Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI) and many others have been adopted by software organisations, but these intervention strategies have not completely addressed the problems faced by software development organisations, especially small software development organisations. Richardson and von Wangenheim (2007) states that small organisations are not familiar with the frameworks and that they perceive them as time consuming and expensive thus difficult to implement. Al Tarawneh, Abdullah and Ali (n.d.) are of the view that these frameworks were created for large software organisations and therefore do not accommodate small organisations, which makes them not

viable in small companies. Romana *et al*., (as cited in Saranya & Kannan, 2013) state that SPI frameworks have not worked in software organisations because of cost, human, and customer communication difficulties. Apart from these frameworks, agile methods have been adopted by software development organisations to address the issue of software project failures. Ferreira and Cohen (2008, p. 48) state that the high software development failure rate can be attributed to traditional software methodologies (the waterfall method). Their words are echoed by Maher, Kourik and Chookittikul (2010, p. 300) who state that the flexibility of agile methods aims to address the problems faced by the software industry such as changing delayed deadlines, cancellations, and applications failing to match customer requirement.

The continuing high failure rate clearly indicates that agile methods are not effective at all. As a result, the software development industry is adopting knowledge management strategies in an attempt to eliminate software development failure problems (Sholla & Nazari, 2011). Software development is a knowledge intensive task which has benefited from knowledge management principles. Knowledge management has brought about many benefits to the software industry. They include increased delivery speed and precision of quality products and it has improved organisational routines and processes (Chandani, Neeraja & Sreedevi, 2007; Bjornson & Dingsoyr, 2008; Gendreau & Robillard, 2013). Silverman (2006) found that knowledge management improves the software development process. The software development process relies mostly on the innovation and experience (knowledge) of workers.

1.2. Statement of the problem

Globally, software development organisations are faced with many challenges, especially software development failures. South African software development organisations face similar challenges to those of their global counterparts (Bogopa, 2010; Gruner & van Zyl, 2011; The Johannesburg Centre for Software Engineering Source Code Report, 2012). There are many reasons for these failures as indicated in the literature (Cerpa & Verner, 2009; Lehtinen *et al*., 2014). Despite many attempts including SPI frameworks and new software development methodologies to address the challenges facing software development organisations, little success has been achieved (Al Tarawneh, Abdullah, & Ali, 2009; Richardson & von Wangenheim 2007; Saranya & Kannan 2013).

To address these issues, software development organisations are now increasingly turning to knowledge management. This is because software development is a knowledge intensive task and knowledge management is contributing to the improvement of software development (Silverman, 2006; Gendreau & Robillard, 2013). Vasanthapriyan, Tian, and Jianwen Xiang (2015, p. 237) state that "knowledge management for software engineering aims at facilitating knowledge flow and utilisation across every phase of a software engineering process.

Globally, studies have shown that software organisations have adopted knowledge management initiatives and its benefits are starting to show (Dingsøyr, 2005; Basili *et al*., 2007; García *et al.*, 2011, and others). In South Africa, it is not known whether software development organisations have adopted knowledge management or not. The literature indicates that this issue is largely being ignored because only three studies (O'Neil, 2005; Silverman, 2006; Khoza & Pretorius, 2016) have been found to be investigating knowledge management issues in the South African context. None of the mentioned studies investigated

knowledge management practices from the knowledge management lifecycle perspective. This has denied the South African software development industry valuable information about knowledge management and its potential benefits.

1.3. Aim of the study

This study therefore sought to understand the nature of software development failures experienced by SMMEs software development enterprises in South Africa with a view of exploring how KM could be used to alleviate the failures.

1.4. Research questions

Specifically, the study answered the following questions:

1. What is the form and source of software development failures experienced by software development SMMEs?
2. What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
3. What benefits has knowledge management brought to software developers and organisations?
4. What knowledge management challenges do software development SMMEs face?

1.5. Significance of the study

The software development process is knowledge intensive (Bjornson & Dingsoyr, 2008; Chandani, Neeraja & Sreedevi, 2007; Gendreau & Robillard, 2013). This means that the whole software development process relies on the knowledge, skills and experiences of the role players, especially the developers. Software development organisations themselves are knowledge intensive organisations. They rely on their knowledge to stay competitive. Globally, the literature has indicated that software organisations are adopting knowledge management and are benefiting from it. In the South African context, that was previously not

known. For example, before this study, little was known about knowledge management practices in small software development organisations. The study has shed light on the status of knowledge management practices, their benefits and challenges in the organisations and in the South African context. This will help the organisations and the entire software development industry find out where they stand and what to improve. The study's recommendations are valuable in that regard. The study has also contributed to the sparse body of literature in this field especially in the South African context. The belief is that it has laid a foundation for many more studies to come as suggested in the recommendations for further research.

1.6. Delimitation of the study

The study was limited to software SMMEs in the KwaZulu-Natal Province of South Africa. The main limitation of the study is that it involved only eight software organisations. However, as for the purpose of most qualitative studies, this was adequate.

1.7. Preliminary literature

This subsection discusses the preliminary literature of the study.

1.7.1. Small, medium and micro enterprises (SMMEs) in South Africa

Small and Medium Enterprises (SMEs) or SMMEs contribute substantially to the gross domestic product (GDP) of many developing and developed countries. Peters and Naicker (2013, p.13) state that SMMEs comprise the majority of enterprises in South Africa and that the government has identified this sector as having the potential to improve job creation opportunities, reduce poverty and create an environment of equitable distribution of wealth. Citing the Small Enterprise Development Agency (SEDA), the Government Gazette (2017, p.16) states that there are 2 251 821 SMMEs in South Africa.

There is no clear definition of an SME or SMME and in South Africa these two terms are used interchangeably although the official government term is SMME. Definitions are influenced by different contexts. For example, the European Union Commission (as cited in Sertić, Barčić & Klarić, 2018, p. 523) defines them as enterprises with fewer than 250 employees or with an annual turnover of up to 50 million Euros or a balance sheet total of no more than EUR 43 million. In the United States of America (USA), an SMME is defined as an organisation with less than 500 employees (United States International Trade commission, 2010, p. 1-3).

The Organisation for Economic Cooperation and Development (OECD) (2017) defines them as firms employing up to 249 persons and breaks them down into micro (employing one to nine people), small (employing 10 to 49 people) and medium (employing 50-249 people). In South Africa, the Department of Trade and Industry (n.d.) defines an SMME depending on the sector of the South African economy. It is defined as an organisation with fewer than 200 full-time paid employees. According to the South African Government Gazette (2017) the term SMME represents small, micro and medium enterprises. There are five categories of SMME in South Africa: survivalist (usually found in the informal sector of the economy), micro (employs between one and five people), very small enterprise (usually employs fewer than 10 people), small enterprise (employs fewer than 100 people), medium (employs over 100 people) (Government Gazette, 2017, p. 14-15).

The Government Gazette further defines ICT SMMEs as "SMMEs that are operating within the ICT sector, either as ICT service providers, software and content developers or electronics and hardware manufacturers" (p.15). The Johannesburg Centre for Software Engineering Source Code Report (2012) states that most software organisations in South Africa are SMMEs.

This study adopted the definitions provided by the Department of Trade and Industry (n.d) and the South Africa Government Gazzette (20017) in that it views SMMEs as organisations with less than 200 fulltime employees.

The importance of SMMEs in the global economy has been well documented. The OECD (2017) lists a number of roles that SMMEs play in global economies. Among them are the contributions to job creation and other social needs and their contribution to national GDPs. In South Africa, SMMEs make up 40% of all businesses and contribute 36% to the South African economy (Smit, 2017). According to Maggs (2010, p.5) South African SMMEs contribute 80% of the jobs and they "stimulate enterprises and encourage productivity gains through pro-competition effects". This assertion is confirmed by Swart (2010, p. 10) who states that small businesses contribute 80% of employment and 30% of the GDP. The latest figures by The Baking Association of South Africa (2018, n.p.) indicate that SMMEs "make up 91% of formalised businesses, provide employment to about 60% of the labour force and have a total economic output which accounts for roughly 34% of GDP" and that they foster diversification through the development of unsaturated sectors of the economy. It further states that ICT SMMEs have the potential to open new business frontiers in developing economies. The Bureau of Economic Research (2016) concurs and states that SMMEs play a significant role in job creation, economic growth, innovation and contribute significantly to the GDP. The Government Gazette indicates that the SMMEs account for 14% of total employment and 42% of the South African GDP. These statistics show that SMMEs play a very crucial role in the economy; therefore, their existence is crucial.

Apart from the contribution that SMMEs make to economies in the developed and developing world, there are a number of challenges that they face. In South Africa, small businesses face challenges in attracting customers, maintaining profitability, increasing revenue, frequent uncertainty over economic conditions, and sourcing finance for expansion

(Business Tech, 2017). Peters and Naicker (2013, p. 21) list corruption, tough competition, lack of access to finance, black economic empowerment (BEE), lack of government support and red-tape, and low growth in respective sectors as the main challenges encountered by SMMEs in South Africa. Retief (2010, p. 14) mentions legislation, lack of access to funding, managing staff, customer complaints (to name but a few) as the main challenges facing SMMEs. The Baking Association of South Africa (2018, n.p.) states that the chief obstacles are crime and corruption, a lack of appropriate technology and low production capacity (including access to electricity), a lack of management skills and inadequate skilled labour, difficulty obtaining finance and credit, little access to markets and developing relationships with customers, recognition by large companies and government bureaucracy, knowledge and support for the role that they play in economic development, and regulatory. The Bureau of Economic Research (2016, p.1) found that SMMEs are "challenged by access to finance and markets, poor infrastructure, labour laws, crime, skills shortages and inefficient bureaucracy".

In other developing countries challenges facing SMMEs have been identified too. Farsi and Toghraee (2014, p.1) list six challenges that SMMEs face. They are research and development; lack of market information, national policy and regulatory environment and managerial, technologies, and human resources. Yoshino and Taghizadeh-Hesary (2016, n.p) state that SMMEs face a lack of databases, limited access to finance, undeveloped sales channels, low R&D expenditures, and low levels of financial inclusion. Bilal and Al Mqbali (2015) found that SMMEs encounter financial, regulatory, environmental and owner capability challenges. Abor and Quartey (2010) found that in South Africa and Ghana, SMMEs face financial constraints, regulatory controls, limited access to international markets, lack of access to existing technology and lack of skills. Mutula and van Brakel (2006) found that SMMEs do not fully utilise Information and communication technologies

(ICTs) to access international markets. SMMEs in developing countries face similar challenges. Common among them are a lack of skills and lack of market information, which are knowledge management related.

1.7.2. South African software development sector

Information communication technologies (ICTs) play a significant role in economic development. ICTs are used all over the world for a number of economic, social, political and many other human activities (Bollou & Ngwenyama, 2008; Zanello & Maassen, 2011). In Africa, ICTs have been used for mainly for economic benefits and communication purposes. The use of mobile phones for financial transactions in East Africa is a classic example. According to Masinde (2016), approximately 27 million Kenyans have mobile money (M-Pesa) accounts. Ndiaye (n.d.) states that nearly 70% of Kenyan adults use the service. Arinloy *et al*. (2015) present cases of mobile phone usages by farmers in Nigeria and Benin. They found that mobile phone messaging reduced market prices. In spite of the positive contribution of ICTs to development, African countries still have a low ICT development index (Ponelis & Holmner, 2015, p. 2).

South Africa has the biggest Information Technology (IT) market in Africa. It is ranked 20th globally (FDI Intelligence, 2012; Switzerland Global Enterprise, 2014). The South African IT industry consists of hardware and software products and services. The Gauteng province is the leader in the IT industry with 60% of all IT projects taking place there; followed by the Western Cape Province (35%) and the KwaZulu-Natal Province (KZN) (5%) (FDI Intelligence, 2012). According to the Independent Communications Authority of South Africa (ICASA) (2016), the South African ICT sector consists of three broad industries; the telecommunications sector, television and broadcasting and postal services. According to Statistics South Africa (2017), the ICT industry contributed 3.1% and 3% to the GDP in the

years 2013 and 2014 respectively, with exports amounting to R 33 794 million Rand in 2014. Schofield (as cited in Kneale, 2017, p. 6) states that the estimated market value of the ICT sector is R270bn or 6% of the GDP. Of this, 66% represents the telecommunications products and services; 16% are computer related services; 8% is hardware; and 8% is software.

Software development falls under the computer programming, consultancy and related activities sector. The Johannesburg Centre for Software Engineering Source Code Report (2012) states that the software industry in South Africa has less than 8000 companies. Most companies are small and they employ less than 50 people. The industry employs about 180,000 people of which 15,000 are software developers. It is estimated that the industry exports $70 million worth of software products and services to the rest of Africa and Europe. It is dominated by five vendors. They are CA Technologies, International Business Machines (IBM), Microsoft, Softline and System Analysis and Programming Development (SAP) who occupy about 40% of the market (Johannesburg Centre for Software Engineering Source Code Report, 2012). The software industry deals mainly with software development and packaged software services. The software industry was projected to be worth $3919 million in 2016.

1.7.3. Knowledge management in software organisations

Knowledge management is a series of processes that include controlling routines and processes, motivating and organising in an organisation to ensure that knowledge is accessed by everyone those who requires it. It involves the creation, storage, transfer, and application of knowledge within the organisation (King, 2009). Knowledge has become an important economic asset. It has emerged as the most strategically significant resource in organisations (Bhandar, Shan-Ling, & Tan, 2005). The main purposes of knowledge management are to

identify and acquire critical knowledge and store, share and apply that knowledge to appropriate situations. It aims at making available knowledge to the right processes at the right times in the right presentation for the right cost (Allard, 2003; Addo & Jennex, 2005).

As already pointed out, software organisations have turned to knowledge management to try to fix the challenges they face (Lindvall, Rus, & Sinha, 2002; Schneider, 2009; Sholla & Nazari, 2011). Knowledge management practices have been adopted in software organisations to manage information and knowledge using different techniques and technologies such as codifying knowledge and building knowledge databases, promoting networks of experts within the organisation, thus enhancing software development processes (Bjornson & Dingsoyr, 2008; Gendreau & Robillard, 2013).

A review of the literature indicates that studies investigating knowledge management in software organisations have been conducted globally, but such studies are conducted mostly in developed countries and in large software development organisations. There are, however, very few studies that have been conducted in developing countries such as Brazil, India, China and South Africa. Studies in this field seem to focus on investigating the role of knowledge management in software development processes (Jelínek, 2010; Kristjansson, Helms, & Brinkkemper, 2014; Dingsøyr & Smite, 2014), knowledge management and architectural software development (Clerc, Lago & van Vliet, 2011; Beecham & Mistrik, 2010), knowledge management and risk in software development (Neves *et al.*, 2014; Alhawari *et al.*, 2012), knowledge management and software testing (de Souza, Falbo & Vijaykumar, 2014; Abdullah, Eri & Talib, 2011), knowledge management in software development teams (Mathrani, Parsons & Mathrani, 2012; Dorairaj, Noble & Malik, 2012) and knowledge management for learning in software development (Boden, Nett & Wulf, 2010; Menolli, Malucelli, & Reinehr 2011; Chouseinoglou *et al.*, 2013).

For example, Jelínek (2010) discussed the application of knowledge management in three software development methodologies, which are the waterfall, agile and the rational unified process (RUP). Dingsøyr and Smite (2014) discuss knowledge management activities in global software development (GSD). Their study revealed three knowledge management activities: knowledge storage, transfer and mapping.

Beecham and Mistrik (2010) identify three knowledge management strategies in global teams, knowledge codification, personalisation and a hybrid of the two. Clerc, Lago and Vliet (2011) found that global agile teams manage architectural knowledge by sharing it across sites through face-to-face meetings and using technology, writing down architectural rules, installing a directory of experts, and creating a repository of architectural artefacts.

Neves *et al.* (2014) investigated knowledge management practices that could minimise risk in software development. They found three major knowledge management activities to analyse risk; conducting meetings and training sessions and creating and using knowledge repositories. Alhawari *et al.* (2012) proposed a knowledge based risk management framework that illustrates the role of knowledge management processes in enhancing and facilitating risk identification, analysis, risk response planning and execution processes in IT projects.

Abdullah, Eri and Talib (2011) developed a knowledge management system model for a community of practice in a software testing environment. Their model is based on the knowledge management lifecycle (create, capture, refine, store, manage, and disseminate). They concluded that this model is a very important feature for the community of practice to manage knowledge in the software testing environment.

Dorairaj, Noble and Malik (2012) describe in detail knowledge management practices in agile teams. They state that agile teams generate knowledge through project inception practices (brainstorming about the project), customer collaboration, formal training,

communities of practice, and self-learning. Mathrani, Parsons and Mathrani (2012) investigated knowledge management practices in offshore software development teams. Their study revealed that software development teams value the need to create knowledge and disseminate it within the organisation.

Boden, Nett and Wulf (2010) investigated organisational learning in offshore teams. The teams were located in different parts of the world with different development cultures. They found that the teams learned from each other's development skills through personal meetings at different sites where developers discussed and shared development ideas. Chouseinoglou *et al.*'s (2013) literature review discusses major and core process areas of organisational learning in software organisations. The major process areas are obtaining, using and passing knowledge.

In South Africa, Silverman (2006) investigated the role of knowledge management in software development. Silverman found that knowledge management practices contributed positively to software development success. O'Neil (2005) investigated how tacit knowledge could be managed in a high technology organisation. O'Neil found that it is a challenge to manage tacit knowledge in software organisations. Khoza and Pretorius (2016) found that poor communication, lack of time, unrealistic expectations, job security and psychological factors affect knowledge sharing in software organisations.

Theoretical papers have also been written on knowledge management in software organisations. Most have been written by researchers in developed countries. A few address issues affecting developing countries. These papers are literature reviews and theoretical frameworks/models (Andrade *et al*., 2006; Bjornson & Dingsoyr, 2008; Chandani, Neeraja & Sreedevi, 2007; Nawina, 2011; Sholla & Nazari, 2011) and many others. Several books (Arum *et al*., 2003; Schneider, 2009) have been published on knowledge management in

software organisations. These books report on how different methods and tools are used for knowledge management in software organisations.

A review of the literature reveals that most knowledge management and software development studies have been conducted in large organisations in developed countries. Little has been done in developing countries such as South Africa compared to other developing countries in the Brazil, Russia, India, China, South Africa zone. The main focus of these studies is the role of knowledge management in software development processes, knowledge management and architectural software development, knowledge management and risk in software development, knowledge management and software testing, knowledge management in software development teams, and knowledge management for learning in software development. Using technology for knowledge management dominates these studies.

This study is different from other studies conducted before in two ways: it uses a different knowledge management lifecycle conceptual framework and it focuses on small software development organisations in South Africa. The conceptual framework was developed for the application in this study. It has not been used in any study before.

1.8. Conceptual and theoretical framework

To answer the questions of the study, a conceptual and theoretical framework was developed. This conceptual and theoretical framework is developed from the software engineering and knowledge management fields. This section briefly discusses the conceptual and theoretical framework informing the study. Details of the framework are found in Chapter Two.

Knowledge management is defined by Aurum, Daneshgar and Ward (2008, p. 515) as an "integrated process incorporating a set of activities in order to create, store, transfer and apply knowledge in the organisation and to add value by using appropriate technologies and

cultural environments". Dalkir (2011) and King (2009) also state that knowledge management involves activities such as knowledge creation, storage, transfer and application. The main goal of knowledge management is to improve organisational routines and processes for a competitive advantage. There are three main knowledge management perspectives found in the literature. They are the codification and personalisation strategy (Hansen, Nohria, & Tierney, 1999), knowledge management schools of thought (Earl, 2001) and the knowledge management lifecycle (Wiig, 1993; Sağsan & Zorlu, 2010), and others. The knowledge management lifecycle perspective views knowledge management as a series of processes. The processes include knowledge acquisition, application, sharing, organisation, and storage. This study adopted the knowledge management lifecycle perspective. The conceptual framework is briefly discussed below.

1.8.1. Knowledge management lifecycle perspective

There are many knowledge management lifecycle frameworks that have been developed over the years. They all provide processes that are believed to be knowledge management activities in organisations. Examples are Wiig (1993), Alavi and Leidner (2001), Sağsan (2006, 2007, 2009), and Evans and Ali (2013). For the purposes of the study, an analysis of 15 existing frameworks was conducted to develop a conceptual framework. The analysis revealed six major knowledge management processes, which are knowledge acquisition, creation, organisation, storage, sharing, and application. These processes form the knowledge management perspective of the study. That is, for the purposes of this study, knowledge management is regarded as a series of six processes.

1.8.2. Learning software organisations

Knowledge management is associated with organisational learning and learning organisations. Organisations that practice knowledge management are called learning organisations (King, 2009). According to Fiol and Lyles (1985, p.803), learning is "a process

of improving actions through better knowledge and understanding". Organisational learning is defined by Oertenblad (2001, p. 126) as the activity of learning in an organisation. Argyris (1977, p.116) states that it is a process of detecting and correcting error, error being any knowledge or knowing that prevents learning. Armstrong and Foley (2003, p.74) define the learning organisation as an organisation that promotes the learning of its individual members to create valued results. Senge (1990, p.13) asserts that a learning organisation is "a place where people are continually discovering how they create their reality and how they change it". Senge further states that in such organisations, people keep on improving their ability to create their truly desired results.

The concept of organisational learning was later adopted by software developing organisations. This is because software development organisations are knowledge intensive. Software organisations that have adopted the learning concepts are called learning software organisations. Chouseinoglou and Bilgen (2012, p.252) define learning software organisations as organisations that learn within the domain of software development. Birk, Dingsøyr, Lindstaedt, and Schneider (2006, p.61) are of the view that it is "an organisation that develops or maintains software and intentionally acts as a learning organisation". Learning software organisations have been created to foster learning at organisational level and to capitalise on the knowledge assets of the software organisation (Althoff, Bomarius, & Tautz, 2000). Feldmann and Althoff (2001, p.2) state that a learning software organisation promotes the management of its knowledge, turns it into intellectual property and then turns it into market shares and profits. They further state that this kind of organisation continuously fosters learning and sharing of experiences within it. This is a kind of an organisation that uses knowledge management to promote learning. By their very nature, software development organisations are knowledge intensive. They rely on the knowledge at their

disposal to develop software, hence they are called learning software organisations. This study treated all software development organisations as learning software organisations.

1.8.3. Software development failures

The second part of the conceptual and theoretical framework is developed from the software engineering literature. It focuses on software development failures and their causes. Linberg (1999) describes a failed project as a project that is terminated before its completion. According to Lyytinen and Hirschheim (1987) a system fails if it does not meet the expectations of the stakeholder. Heeks (2002) categorises failure in two classes: total failure – where a system is never implemented, or implemented but immediately abandoned; a partial failure – where most of the outcomes of the systems are not met. The Standish Group's CHAOS Manifesto (2013, p.1) classifies project failures into challenged and failed. A challenged project is "late, over budget, and/or with fewer than the required features and functions". A failed project is "cancelled prior to completion or delivered and never used". There are many causes of software development failure. They include the following: methods used to develop the system, environmental and task factors, planning, budget and risk, executive support, sponsorship and inadequate management structure, contractor and stakeholder relationship, project complexity, staff turnover, commitment, objectives and evaluation stage and inadequate post-mortem processes (Ewusi-Mensah & Przasnyski, 1991; Cerpa & Verner, 2009; Lehtinen *et al*., 2014; Hughes, Rana, & Simintiras, 2017).

For the purposes of this study, a software project failure is defined according to Lyytinen and Hirschheim (1987) and categorised according to Heeks (2002) and the Standish Group's CHAOS Manifesto (2013). A failure is stated to be a system that does not meet the client's expectations because it is either challenged and/or abandoned.

Table 1 shows the conceptual and theoretical framework of the study and the research questions it is going to answer.

Table 1: conceptual and theoretical framework

Knowledge Management concepts	
Concept	**Research question addressed**
Acquisition	RQ 2 - What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Application	RQ 2 - What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Creation	RQ 2 - What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Organisation	RQ 2 - What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Sharing	RQ 2 - What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Storage	RQ 2 - What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Software development failure concepts	
Failure	RQ 1 – What is the form and source of software development failures experienced by software development SMMEs?
Types of failure	RQ 1 - What is the form and source of software development failures experienced by software development SMMEs?
Causes of failure	RQ 1 - What is the form and source of software development failures experienced by software development SMMEs?

Research questions three (What benefits has knowledge management brought to software developers and organisations?) and four (What knowledge management challenges do software development SMMEs face?) fall out of this conceptual and theoretical framework but they addressed knowledge management issues, hence their inclusion in the study.

1.9. Research methodology overview

This section discusses the research design of the study. It discusses the philosophical perspective, the methodology and methods, ethical issues, reasoning and issues of validity and reliability. First to be discussed will be philosophical perspectives, that is, the ontological and epistemological perspectives.

1.9.1. Ontological perspectives

Ontological perspectives define the real world, whether physical or abstract. Ontology is defined by Marsh and Furlong (2002) as the theory of being. Ontology defines what we human beings define as our environment, our universe and the way we live in it. Realism and relativism are the main ontological assumptions in research. Realism is a view that supports the existence of a real, physical world independent of individuals and human experience (Schuh & Barab, 2008). Relativism assumes that reality exits through social interactions and experience with the environment. This study adopted relativism because relativism assumes that knowledge is socially constructed between the interaction of people and the environment they live in. In organisations people create reality by interacting with the organisation (environment). Knowledge management is regarded as a socially constructed phenomenon.

1.9.2. Epistemological perspectives

Epistemology is how we assume knowledge is created. Research paradigms are used to explain how knowledge is created. There are many research paradigms that explain how knowledge is constructed. Hirschheim and Klein (1989) identifies four such research paradigms: functionalism, social relativism, radical structuralism, and radical humanism. Olikowsky and Baroudi (1991) identify three main paradigms used in social research: the positivist, interpretive, and critical paradigms. Adopting Deetz's (1996) framework, Schultzer and Leidner (2002) identify four scientific discourses to study knowledge management in organisations. They are the normative, the critical, the interpretive, and the dialogic

discourses (Schultzer & Leidner, 2002). Chua (1986) identifies four paradigms used to inform research: the functionalist, interpretive, radical humanist, and radical structuralist paradigms.

The functionalist paradigm explains the current social order, social integration, consensus, need satisfaction, and rational choice and further explains how individual elements of a social system interact to form an integrated whole (Hirschheim & Klein, 1989, p. 1201). According to Olikowski & Baroudi (1991, p. 5) interpretivism assumes that human beings develop and associate their own subjective and inter-subjective meanings as they interact with their surroundings. Interpretivism does not define dependent and independent variables (Klein & Myers, 1999). Critical theory is a form of historical materialism and is influenced by issues of class, ethnicity, and gender. It views situations through a lens of local domination by powers with the potential for local resistance (Brooke, 2002). Specific causes such as feminism and environmentalism usually influence critical researchers (McGrath, 2005). Positivistic research tests theory. It features formal propositions, quantifiable measures of variables, hypothesis testing, and the drawing of inferences about phenomena from a sampled population (Olikowski & Baroudi, 1991). It is argued that most (if not all) of these paradigms emanate from the Burrel and Morgan framework (Burrell & Morgan, 1979) which identifies four paradigms: functionalist, interpretive, humanist and structuralist paradigms.

This study adopted the interpretive paradigm. Interpretive studies have gained considerable attention in recent years as an alternative to positivist studies (Lee, 1991). Interpretive research assumes that our knowledge of reality is gained through social constructions. Social constructions can be consciousness, language, shared meaning, and other artefacts. This study applied interpretivism because knowledge management is a social process that occurs in a social context. Interpretivism was adopted because of its relationship with relativism and the fact that both do not separate the person from the environment and because of their emphasis

on a socially constructed world and knowledge, and on the subjective nature of knowledge. This study assumes that knowledge management is a subjective concept which comes into being through the interpretation of actions of actors. It also assumes that knowledge is socially constructed. One has obtained knowledge after interacting with other people and objects in the environment they live in.

1.9.3. Research methodology

Three research methodologies are commonly used in social research: qualitative, quantitative and a combination of the two (sometimes referred to as mixed or multiple methods) (Creswell, 2009). Qualitative research involves collecting data in participants' natural setting, analysing the data inductively, and making the interpretations of the meaning of the data (Creswell, 2009). Quantitative research involves testing hypotheses and the development of theory which could be generalised across settings (Amaratunga *et al*., 2002). The mixed or multiple methods entail combining qualitative and quantitative methods.

This study adopted the qualitative research methodology. This is because a qualitative approach enabled the respondents to express their views without being given options to choose from and to divert from the quantitative approaches mostly used to investigate project failures such as the Standish Group and ITWeb. Interviews were adopted as a qualitative approach.

1.9.4. Research method

The choice of a method depends on the research question(s) of the study. This study adopted the multiple case study method. According to Yin (2014, p.16), a case study "investigates a contemporary phenomenon in depth and within its real-world context especially when the boundaries between phenomenon and context may not be clearly evident". As a method of enquiry, Yin (2014, p.17) defines a case study as a study that "copes with technically

distinctive situations in which there will be many more variables of interest than data points and relies on multiple sources of evidence with data needing to converge in a triangulation fashion. It benefits from the prior development of theoretical propositions to guide data collection and analysis". Creswell (2009) states that in a case study, the researcher explores a program, organisation, event or activity in-depth. Benbasat, Goldstein and Mead (1987) provide three reasons why case study research is used: 1) the researcher can study a phenomenon in its natural settings, 2) It allows the researcher to ask 'how' and 'what' questions, and 3) a case study is suitable where a few previous investigations have been conducted.

A multiple case study research analyses findings of more than one case to yield greater robustness. It provides stronger evidence that the results of the study can be generalised (Valentino, 2017, p.1-2). This is confirmed by Saunders, Lewis and Thornhill (2009), who state that the main purpose of a multiple case study is to establish if the results of the first case occur in the other cases. Green and David (1984, p.75) define a multiple case study as a research strategy for generalising the results of a number of sampled cases. They further present four characteristics of multiple case study research: it must have a conceptual framework, a sampling plan, procedure for conducting single case studies and cross-site analysis.

This was a multiple case study because it involved 12 homogeneous organisations in the province of KwaZulu-Natal (KZN), South Africa. The organisations were investigated in their natural settings and the study asked 'what' questions; multiple sources of data were used.

1.9.5. Time horizons

Two research time horizons are common in research: cross-sectional and longitudinal time horizons (Hussey & Hussey, 1997). Cross-sectional studies collect data just once over a period of time. Longitudinal studies collect data more than once, over a period of time. This study applied the cross-sectional research horizon. Data are collected only once.

1.9.6. Logic of reasoning

Blaike (2010, p. 81) presents four research approaches (logic of reasoning) that are used in social research. They are induction, abduction, deduction and retroductive research. Induction studies develop theory from observation of empirical reality (Collis & Hussey, 2003). Deduction involves the development of a theory which is then subjected to rigorous testing (Saunders, Lewis & Thornhill, 2009, p. 124). Retroductive research aims to "discover underlying mechanisms that explain observed regularities in particular contexts." Abduction involves developing theories derived from social actors, meaning in the context of everyday activities (Blaikie, 2010, p. 89). The study adopted induction. Theory was developed based on data collected and analysed.

1.9.7. Population and sampling

The population of the study was software development organisations based in the KZN province of South Africa. Software development organisations are organisations whose core business is the development, and maintenance of software products. According to Kneale (2017, p.3) software companies in general consist of companies that develop application software products under their own names, start-up software and application developers and vendors who resell, install and upgrade the software, either as licensed resellers or retailers. KZN was selected because it is within the reach of the researcher and because it has the third largest number of IT projects in South Africa (FDI Intelligence, 2012). The study targeted SMMEs. This is because according to the Johannesburg Centre for Software Engineering

Source Code Report (2012) most software development organisations in South Africa are small, medium or micro. Twelve software development organisations participated in the study. The 12 organisations were identified based on a list provided by the Johannesburg Centre for Software Engineering Source Code Report (2012), snowball sampling and web searches. In the 12 organisations, IT/software project managers and software developers were targeted as data sources.

1.9.8. Validity and reliability

Issues of validity and reliability are very important in research. These are benchmarks that are used to validate research. In quantitative studies, issues of validity and reliability are well defined and accepted. That is not the case in qualitative studies and this has caused debate about the validity and reliability of qualitative studies. In qualitative research, validity and reliably is known by a number of terms. For example, Guba and Lincoln (as cited in Morse *et al*., 2002) refer to trustworthiness, transferability and credibility. Sarantakos (as cited in Bapir, 2012) lists cumulative, communicative, argumentative, and ecological validation. This study adopted the notion of reliability and validity in the context of qualitative research. A pilot study was conducted in the Gauteng province to test the instruments. Adjustments were made where necessary before the actual study was undertaken in KZN.

1.9.9. Data collection

In the social sciences, questionnaires, document analysis, interviews and observations are popular data collection instruments. In this study, semi-structured interviews were used to collect data. Interviews were conducted with IT/software project managers and software developers.

1.10. Data analysis

Content analysis was used to analyse data obtained from interviews. According to Krippendorff (2004, p.3), content analysis "is a systematic reading of texts, images and symbolic matter through classification, tabulation and evaluation in order to ascertain its meaning and probable effect". Elo and Kyngäs (2008, p.107) state that content analysis can be quantitative or qualitative and could also be deductive or inductive. This study adopted inductive qualitative content analysis as proposed by Elo and Kyngäs (2008, p. 110). There are three major steps involved in the inductive content analysis. They are preparation, organising and reporting. Data were then categorised into two main themes: software development failures and knowledge management practices and knowledge management benefits and challenges.

1.11. Ethical considerations

Every researcher is expected to consider research ethics as he/she conducts research. In any research involving people, respondents are protected from any form of harm and abuse. The University of KwaZulu-Natal (UKZN) has a code of research ethics that it expects every researcher to conform to. This research was conducted in conformity with the UKZN code of ethics. Permission was sought from participating organisations to conduct the study. Respondents were notified that participation was voluntary and that they could withdraw at any time during the study. Privacy and anonymity of organisations and people were observed. Information obtained from company documents were kept private and were used for the purposes of this study only.

1.12. Contribution to knowledge

The study has made theoretical and practical contributions. Theoretically, the study contributed by developing a conceptual framework from two fields of study, knowledge management and software engineering. This is the first study to have used this framework to investigate knowledge management and software engineering phenomena. The study has also contributed to the existing body of literature of the two fields. Contribution to practice has been made by providing suggestions for improving knowledge management which in turn will improve software development practices in organisations. The study revealed software development failures encountered by software development organisations. The software development industry is expected to use the findings to produce solutions to the software development failure problems.

1.13. Structure of the dissertation

The report consists of seven chapters. Table 2 shows the structure of the thesis.

Table 2: Structure of the report

Chapters	**Description**
Chapter 1: Background and introduction of the study	This chapter gives an introduction of the work and gives background information about the study and the research problem.
Chapter 2: Conceptual and theoretical framework	In this chapter, the concepts used in the study are defined in relation to the context of the study. The theoretical underpinnings of the study are also discussed.
Chapter 3: Literature review	Literature on knowledge management and software engineering is surveyed in this chapter. The focus of this chapter is software development failures and knowledge management in software engineering.
Chapter 4: Research design	The research design of the study is presented in this chapter. This includes the philosophical perspectives, research approaches, population of study, sampling techniques, reliability and validity, data collection methods, data analysis procedures and ethical considerations.
Chapter 5: Data analysis and presentation of findings	Data collected is analysed and presented in this chapter.
Chapter 6: Discussions of findings	The data presented in the last chapter are discussed, comparisons are made and conclusions drawn using literature and theory.
Chapter 7: Summary, conclusions and recommendations	A summary of the findings is presented in this chapter. Conclusions drawn from the study are also presented. Lastly to be presented are the recommendations of the study and areas for future research.

1.12. Summary

The chapter presented background information about the research problem, the aims and objectives of the study. It went further to provide a summarised version of the conceptual and theoretical framework and research design. The research design discussed the ontological and epistemological perspectives, methodology, sampling techniques, the population, research instruments, ethical issues, issues concerning validity and reliability and how the data were analysed. The last part of this section describes the structure of the thesis.

CHAPTER TWO
CONCEPTUAL AND THEORETICAL FRAMEWORK

2.1. Introduction

This chapter presents the conceptual and theoretical framework of the study. It also presents the concepts that are used in the study and their related theories. The conceptual and theoretical framework draws from two related fields: knowledge management and software engineering. This is because the phenomena being investigated by the study cut across the two fields. The chapter discusses knowledge management, its concepts and frameworks. This is followed by software engineering concepts which are software development failure and its types. The concepts organisational learning and learning software organisations are later introduced. This is because the study views software organisations as learning organisations. This is because knowledge intensive organisations are regarded as learning organisations. First to be discussed will be the concepts data, information and knowledge. Defining these concepts will lay the foundation for conceptualising knowledge management.

2.2. Data, information, and knowledge hierarchy

The concepts data, information, and knowledge have to be defined for two reasons. Firstly, they are interrelated, and secondly because they are used interchangeably sometimes. These concepts are confused and used interchangeably as if they have the same meaning when in fact they are different. The relationship between these concepts is blurry, hence the importance of defining them in the context of this study. The discussion of these concepts lays the foundation for the discussion of the concept ‘knowledge management’.

The concept data is defined first, followed by information and lastly knowledge. The data, information, knowledge hierarchies are discussed at the beginning of the section.

2.2.1. Data

The concept data has been defined in many contexts in many different academic fields. Examples are Information Science, Knowledge Management, Computing and others. Data are series of disconnected facts and observations. They are information objects in binary code waiting to be stored in a computer. They consist of information in a narrow sense (Zins, 2007, p. 482). Tiwana (2002) describes data as a set of objective facts about an event or just structured records of a transaction. Tiwana further describes data as meaningless figures, digits, sentences and phrases that cannot be used for decision-making. According to Rumizen (2002) data are bits and numbers, are discrete, have no meaning, are self-contained and exist in isolation. Ahsan and Shah (2006, n.p.) define data as simple facts that could be used to generate information. They further state that data are raw and have no meaning. Data can exist in any form, usable and not usable.

Hey (2004) and Tiwana (2002) indicate that data could be explained differently in different fields and contexts. In the Information Science field, it is defined as unprocessed information (Hey (2004, p. 5). In the Computer Science field, data are streams or pockets of unprocessed digital information. Tiwana (2002) indicates that, in Information Systems, data are regarded as unprocessed information or as computer data stored in binary format. The Information Science perspective given by Hey (2004) is adopted in this study. The study regards data as unprocessed information. The unprocessed information could be documents (hard copy and electronic) and messages from people.

2.2.2. Information

Information is defined as relevant, useful data that could be used for decision-making. Hoppe, Seising, Nurnberger and Wenzel (2011, p. 585) define it as data that has been given meaning. Ahsan and Shah (2006, n.p.) indicate that information is meaningful data. Chaffey and Wood (2005) describe information as data with value to the understanding of a subject and in

context. According to Chaffey and Wood (2005), information is the basis of knowledge. Information is also defined as data that is interpreted into a meaningful framework (Vance as cited in Alavi & Leinder, 2002). Drucker (as cited in Tiwana, 2002, p. 40) describes it as relevant and purposeful data. The review of the literature indicates that most scholars agree that information is processed, meaningful, useful data that can be used for decision-making in different contexts. The study adopted this definition of information.

2.2.3. Knowledge

Knowledge is an abstract notion. Literature gives different definitions of knowledge. According to Mendez, Gomes and Batiz-Lazo (2004), when data and information are fully utilised they become knowledge. The utilisation of data and information must be coupled with the abilities of peoples' motivations, ideas, skills, intuition, competencies, and commitments. They further state that holistically, knowledge is considered to be our perspectives and concepts, root causes, talents, ideas, and judgments. They also state that knowledge is stored in the individual brain or encoded in an organisation's processes, documents, products, services, facilities and systems.

Spender (as cited in Mendes, Gomes & Batiz-Lazo 2004, p. 151) states that "knowledge is conceived as a manageable asset, just as cash flow, raw materials, or human resources." Alavi & Leidner (2001, p. 109) defines knowledge as information possessed in the minds of individuals. It becomes knowledge once the information is articulated and presented in different forms such as, words, works, graphics and text.

Alavi and Leidner (2001) state that there are several perspectives of knowledge. knowledge could be a capability, state of mind, process, object, and a condition of having access to information. Tsai, Chang & Chen (2006, p. 61) state that knowledge includes, intuition,

vision, value, structured experience, and judgement. Zander and Kogut (1995) state that in general, knowledge in a firm can be categorised into know-how and information. They state that knowledge consists of the competences of individuals and the organising of principles by which a relationship among individuals, teams and industrial network members are structured and coordinated. The study adopted Mendez, Gomes and Batiz-Lazo's (2004) definition. This is because this definition is very broad. It includes aspects such as documents, products and services. In a software development organisation, knowledge could be documents, software products and peoples' skills (services).

There are two opposing hierarchical structures that explain how knowledge is generated. One structure assumes that knowledge is generated from data, and information. This structure is supported by Bierly, Kessler and Christensen (2000). The other structure assumes that knowledge does not come from either data or information, but that data and information are created from knowledge. This notion is supported by Braganza (2004). Bierly, Kessler and Christensen (2000) argues that data is acquired first and then processed to information and finally knowledge. On the other side Braganza (2004) argues that knowledge has always been available. Therefore, people use their knowledge to generate information and data. Figures 1 and 2 show the data, information and knowledge approach.

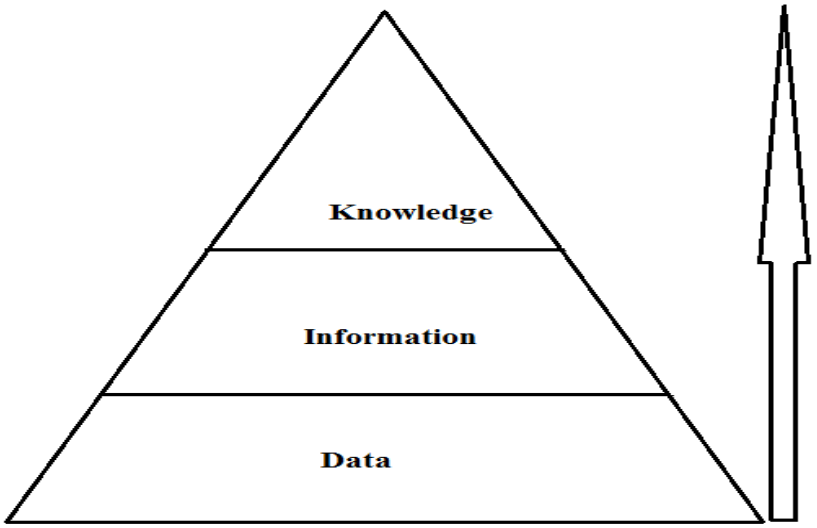


Figure 1: data, information knowledge (DIK) hierarchy (Source: adapted from Braganza, 2004, p. 348)

Figure 2 shows the knowledge, information and data approach.

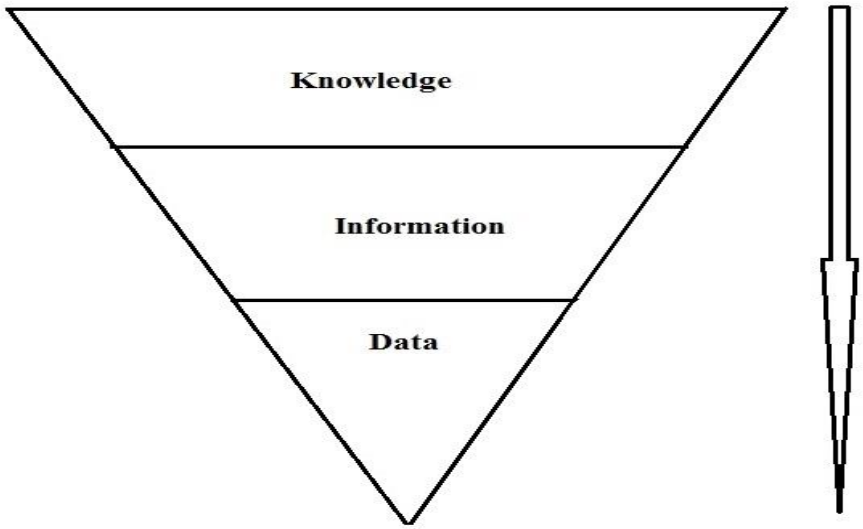


Figure 2: Knowledge, information and data (KID) hierarchy (Source: adapted from Braganza, 2004, p. 245)

Figure 1 shows that knowledge is derived from data and information and figure 2 indicates that knowledge is not derived from data and information, but it is information and data that are derived from knowledge. In short, the figures show the different approaches to the existence of knowledge.

The study adopted the data, information, and knowledge approach. This is because this approach is widely accepted in many fields of study and in the field of Information Science in particular.

2.2.3.1. Types of knowledge

There are many types of knowledge given in the literature, but the widely adopted types are tacit and explicit knowledge. Polanyi (1962) is credited for being the first in differentiating knowledge. The following subsection will discuss knowledge types focusing mainly on tacit and explicit knowledge.

2.2.3.1.1. Tacit knowledge

Drawing from Polanyi, Nonaka and Takeuchi (1995, p. 59) define tacit knowledge as personal, context specific knowledge that is difficult to share. Marwick (2001, p. 814) agrees with Nonaka and Takeuchi and states that tacit knowledge is what the knower knows. It is knowledge derived from experience and includes values and beliefs. The same view is shared by Gladstone (2000) who states that tacit knowledge is what an individual knows. Gladstone further states that tacit knowledge is personally held and it is sometimes not recognised by the holder that it is knowledge. It includes subjective know-how, insights and intuitions. It is dynamic, and constantly changing with the experiences of its processor (Gladstone, 2000).

Elfving and Funk (2006) are of the view that tacit knowledge is held and dependent on the individual holder. It is the individual's experiences and understanding of a phenomenon or concept. They also agree with the other scholars that it is personal and difficult to formalise, elusive, subjective; thus making it difficult to disseminate. Leonard and Sensiper (1998, p. 115) say that tacit knowledge provides a deep understanding of how something works, thus allowing the individual to anticipate and predict occurrences that are explored consciously.

Two dimensions of tacit knowledge are given by Nonaka Takeuchi (1995), the technical and cognitive dimension. The technical dimension encompasses informal personal skills and craft, and referred to as know-how. The latter consists of beliefs, values, mental models and ideals which are deeply ingrained in human beings and are often unnoticed or taken for granted.

An analysis of the literature indicates that tacit knowledge is knowledge that is hidden in peoples' heads, thus making it difficult to notice that it exits, and difficult to communicate and share with others. The literature (Marwick, 2001; Elfving & Funk, 2006) gives almost similar definitions of tacit knowledge, but Nonaka and Takeuchi's definition is adopted in the study. This is because it goes further to include the technical and cognitive dimensions of tacit knowledge. That is the tacit knowledge that is found in software organisations. The technical dimension is the software development skills that the developers have. That is the ability to create a software product. The cognitive dimension is the ability to conceptualise, understand and turn a business problem into a software product.

2.2.3.1.2. Explicit knowledge

Nonaka and Takeuchi (1995) define explicit knowledge as knowledge represented by some artefact, such as a document or a video, which has typically been created with the goal of communicating with another person. Gladstone (2000) on the other hand describes explicit knowledge as knowledge held formally in the form of reports, equations, formulae and specifications. It is easily transmitted between individuals and teams. Nonaka (1997) says that explicit knowledge is objective and rational that can be expanded in words and writings. Explicit knowledge can be codified and shared in face-to-face interactions (Sambamurthy & Subramani, 2005; Rice & Rice, 2005).

Scholars (Rice & Rice, 2005; Sambamurthy & Subramani, 2005) indicate that explicit knowledge is knowledge that has been documented and can be easily shared. It is knowledge that can be stored in a number of sources like paper and electronic databases. It is knowledge in the form of an object. In software development, that could be the software product itself, the documents used to develop the system or the system documentation or databases of best practices and lessons learned. The definitions of explicit knowledge are very similar. One finds it difficult to choose a specific one. The study will adopt all the definitions given above.

There is a lot of debate about the importance of tacit knowledge and explicit knowledge. Researchers such as Nonaka and Polanyi emphasise that tacit knowledge is the most important. They are of the opinion that explicit knowledge is derived from tacit knowledge hence its importance. Researchers such as Tsoukas (1996) say that separating the two types of knowledge is missing the point. They are complementary. The study supports Tsoukas' argument. In software development, both types of knowledge are important. The software product (explicit knowledge) is a product of software development processes (tacit knowledge).

2.2.3.1.3. Other types of knowledge

Apart from the two types of knowledge discussed above, other writers have added other types of knowledge to the knowledge literature. For example, Holsapple and Joshi (2002, p.49) have added descriptive, procedural and reasoning knowledge. They state that descriptive knowledge describes the state of the world and define procedural knowledge as concerned with how something is done (for example a process or technique, the software development process in this study). They further state that reasoning knowledge indicates conclusions that one takes in a certain situation (for example cause-and-effect relations).

Blackler (1995, p. 1023-1025) introduces embodied, embrained, embedded, encultured and encoded knowledge. Embodied knowledge is action oriented knowledge or know-how. It is the ability to perform an action. Embrained knowledge is dependent on conceptual skills and cognitive abilities. Blackler calls it knowledge-that. Embedded knowledge resides in systems' routines and processes. Encultured knowledge is knowledge of the processes aimed at achieving a single organisational goal. Encoded knowledge is information conveyed by symbols and signs (Blackler, 1995, p. 1023-1025). Spender (1996, p.68) distinguishes between individual and organisational knowledge. Individual knowledge is held by individuals within the organisation while organisational knowledge is the sum of all the knowledge held by the individuals which is used for the operations of the organisation.

Stenmark (2001) names other researchers such as Choo who has added cultural knowledge. Stenmark says that all these types of knowledge are deduced from Polanyi's notion of tacit knowledge. The study adopted the two widely used types: tacit and explicit knowledge. The study also recognises Spender's (1996) differentiation between individual and organisational knowledge. This is because the study treats the knowledge of software developers as

individual knowledge and the sum of all the knowledge in the organisation as organisational knowledge. The following section discusses knowledge management.

2.3. Knowledge management

There has been a lot of debate in the literature about the concept knowledge management. The concept has created considerable interest among academics and practitioners because of its relationship with innovation and as an important organisational resource (Spender, 2003). Maki, Jarvenpaa and Hamalainen (2001) are of the opinion that the concept knowledge management is broad and ambiguous. Handzic (2001) says that literature shows that the definitions of knowledge management vary in scope and focus.

Gottschalk (2007, p. 10) defines it as organised processes for acquiring, organising, and communicating knowledge to employees so that they can make good use of it. Knowledge facilitates effectiveness and productivity. Davenport and Prusak (as cited in April & Izadi, 2004, p. 17) define knowledge management as "concerned with the exploitation and development of the knowledge assets of an organisation with the view of furthering the organisation's objectives." Dalkir (2011, p. 5) provides a number of definitions by different authors. The definitions provided by Dalkir indicate that knowledge management activities have to do with the capture, transfer and application of knowledge in organisations. King (2009) avers that it involves the storage, creation, transfer, and application of knowledge in organisations.

Knowledge management is also defined as the ability of an organisation to successfully use its "collective wisdom to increase innovation and responsiveness" (Hackbarth, 1998, p. 588). The goals of knowledge management are to identify and acquire critical knowledge, store, share and apply that knowledge to appropriate situations. Knowledge management aims to make available knowledge to the right processes at the right times in the right presentation for

the right cost (Allard, 2003; Addo & Jennex, 2005). Holsapple and Joshi (2002) shares these sentiments and state that the goal of the knowledge management initiative is to make the right knowledge available to the right processes (human or computer) at the right times in the right presentation for the right costs. Aurum, Daneshgar, and Ward (2008, p. 515) state that "knowledge management can be defined as an integrated process incorporating a set of knowledge management activities in order to create, store, transfer and apply knowledge in a knowledge business value chain using appropriate technologies and cultural environments".

Not all scholars agree though on the concept of knowledge management. One such critic is Wilson (2002; 2005). Wilson states that knowledge management is a management fad. He compares knowledge management to other management fads such as business process re-engineering, total quality management, and organisational learning that, according to him, have failed dismally. He writes "knowledge management (whatever it is) shows signs of being offered as a utopian idea which is likely to fail just like the previous fads" (Wilson, 2005, p. 152). Wilson argues that the advocates of knowledge management make no clear, operational distinction between knowledge and information. He further states that knowledge management originates from artificial intelligence and expert systems. He says that there is no core to the literature of knowledge management and that Nonaka and Takeuchi's (1995) distinction between tacit and explicit knowledge illegitimately corrupts Polanyi's idea of tacit knowledge.

Wilson is supported by Gorman (2004, p.1) who states that knowledge management is an oxymoron because he asserts that knowledge cannot be managed. Gorman's argument is based on the premise that tacit knowledge cannot be managed because it belongs to the individual who holds it. Gorman further argues that what is believed to be knowledge is actually information. Therefore, there is no difference between knowledge management and information management except that knowledge managers are paid more than information

managers and librarians. In South Africa, Ocholla and Shongwe (2013) found that information and knowledge managers and librarians are hired mainly in the public sector and that information and knowledge managers are paid higher salaries than librarians (p. 40). Ferguson (2004, p.1) also argues that the knowledge management literature fails to distinguish itself from information management literature.

The study disregarded the critics and accepted that the concept of knowledge management exists. Grant (2015) argues that knowledge management has long passed the fad phase. This is because, according to Grant, knowledge management continues to attract interest from academics and practitioners. The study then adopted the notion that knowledge management promotes innovation and competitive advantage (Spender, 2003) and that it involves the creation, storage, transfer and use of knowledge in organisational routines and processes (Hackbarth, 1998; Dalkir, 2011) and that knowledge management seeks to make knowledge available to those who need it at the right place and time and in the most suitable format (Allard, 2003; Addo & Jennex, 2005).

2.3.1. Knowledge management perspectives

This subsection discusses knowledge management perspectives. The knowledge management literature (Hansen, Nohria & Tierney, 1999; Earl, 2001; Shongwe, 2016a) reveals three knowledge management perspectives widely adopted by organisations. They are the codification and personalisation perspective, knowledge management schools of thought and the knowledge management lifecycle perspective. The three perspectives are discussed below.

2.3.1.1. The personalisation and codification perspective

Hansen, Nohria and Tierney (1999, n.p.) present two approaches to knowledge management. They are the personalisation and codification perspectives. According to Hansen, Nohria and Tierney (1999) codification is when knowledge is codified and stored in repositories for easy access and retrieval. Codification entails codifying knowledge from people to documents. The knowledge is extracted from the person who developed it, made independent from that person, and reused for various purposes. It is further stated by the authors that this strategy allows people to search for and retrieve stored knowledge without having to contact the owner. That opens up the possibility of achieving a high level of reuse and thus growing the business.

On the other hand, the personalisation strategy focuses on dialogue between individuals, but not codifying and storing knowledge. In this approach, knowledge is shared face-to-face in meetings and brainstorming sessions, and one-on-one meetings (Hansen, Nohria & Tierney, 1999, n.p.). The company does not separate the knowledge from the original developer, but creates a network of experts among whom knowledge can be obtained and shared. In short, the codification strategy aims to convert tacit knowledge into explicit knowledge by extracting it from the knower and storing it in explicit form (documents and databases). The personalisation strategy entails keeping the knowledge in the knower's head but encouraging the knower to share it with others.

2.3.1.2. Knowledge management schools of thought

Earl (2001) classifies knowledge management initiatives in organisations according to schools of thought. The schools describe the knowledge management focus that organisations take. There are three broad schools of thought as classified by Earl (2001). They are the technocratic, economic and behavioural schools.

The technocratic school

The technocratic school focuses on information management technologies which support knowledge workers in their daily tasks. It is further divided into systems, cartographic and engineering schools. The systems school focuses on the capture of specialist knowledge and stores it in knowledge databases. It focuses on the development of knowledge repositories and their initial use. The cartographic school focuses on mapping organisational knowledge. The main purpose of this school is to ensure that experts in an organisation are accessible to everyone for advice (Earl, 2001). The engineering school focuses on knowledge management as a branch of business process reengineering. It focuses on knowledge processes (Earl, 2001).

The behavioural school

The behavioural school is concerned mainly with stimulating and orchestrating managers to be more proactive in the creation, sharing and use of knowledge assets. This school is further divided into organisational, spatial, and strategic schools. The behavioural school is concerned with the use of organisational structures to share knowledge. This school focuses on describing the use of structures within the organisation to share knowledge. The spatial school's focus is on encouraging socialisation within the organisation as a means of knowledge sharing. The strategic school "sees knowledge management as a dimension of competitive strategy" (Earl, 2001, p.227). It is concerned with the development of conceptual models of the nature of intellectual capital.

The economic school

This school is mainly concerned with the commercial aspects of knowledge. It is concerned with converting knowledge or intellectual capital into commercial value (profits) (Earl, 2001, p. 222). Examples are companies that deal with the creation and management of intellectual property such as patents and trademarks.

2.3.1.3. The knowledge management lifecycle perspective

The literature defines knowledge management mostly from the lifecycle perspective. This is a series of processes that it is assumed organisations employ when managing their knowledge resources. The knowledge management literature provides many lifecycle frameworks with many different processes. Widely accepted processes found in these frameworks are knowledge creation, storage, sharing, and application. Sağsan and Zorlu (2010, p. 406) drew up a list of 13 knowledge management frameworks. Heisig (2009) compares 160 such frameworks in his paper on harmonising knowledge management.

This subsection discusses and analyses 15 knowledge management lifecycle frameworks to develop a unified framework adopted in this study. Of the 15 frameworks, 13 were given by Sağsan and Zorlu (2010, p. 406). An additional two by Huber (1991) and Evans and Ali (2013) are added to the list. Huber's (1991) framework has been added because of its wide adoption and Evans and Ali's (2013) framework has been added because it has been developed recently. Table 3 shows the frameworks.

Table 3: Knowledge management lifecycle frameworks

1	**Huber (1991)** Acquisition, distribution, interpretation, organisational memory
2	**Wiig (1993)** Creation, sourcing, compilation, transformation, dissemination, application, value realisation
3	**Meyer and Zack (1996)** Acquisition, refinement, storage/retrieval, distribution, presentation
4	**Nickols (1996)** Acquisition, organisation, specialisation, storage/access, retrieval, distribution, conservation, disposal
5	**Skyrme (1998)** Identify, create, collect/codify, knowledge database, diffuse/use
6	**Bukowitz and Williams (1999)** Get, use, learn, contribute, assess, build/sustain, divest
7	**Alavi and Leidner (2001)** Creation, storage/retrieval, transfer, application
8	**McElroy (2003)** Individual and group learning, knowledge claim validation, information acquisition,
9	**O'Dell, Grayson and Essaides (2003)** Organizing, sharing, adapting, using, creating, defining, collecting
10	**Rollet (2003)** Planning, creating, integrating, organising, transferring, maintaining, assessing
11	**Awad and Ghaziri (2004)** Capturing, organising, refining, transferring
12	**Becerra-Fernandez, Gonzalez and Sabherwal (2004)** Discovery, capture, sharing, application
13	**Dalkir (2011)** Knowledge capture and/or creation, knowledge acquisition and applications, knowledge sharing and dissemination
14	**Sağsan (2006, 2007, 2009)** Knowledge creation, knowledge sharing, knowledge structuring, knowledge using, knowledge auditing
15	**Evans and Ali (2013)** Identify, organize and store, share, apply, evaluate and learn, create

Except for Huber (1991) and Evans and Ali (2013), all other frameworks were adapted from Sağsan and Zorlu (2010:406)

A brief discussion of the different frameworks is presented below.

2.3.1.3. 1. Organisational learning processes (Huber, 1991)

The organisational learning framework explains how learning occurs through knowledge management initiatives in organisations. It has four knowledge management processes that support learning; knowledge acquisition, information distribution, information interpretation, and organisational memory. Huber (1991, p. 90) defines knowledge acquisition as the process

of obtaining knowledge. Huber (1991, p. 91) further states that knowledge could be acquired by reading, conducting customer surveys and research and development activities. Information distribution is the sharing of information from different sources which leads to the development of new knowledge and/or understanding. Information interpretation involves giving meaning to the distributed knowledge. Organisational memory is the storage of information and knowledge for future use. Organisational memory can take two forms, knowledge stored in human heads and knowledge stored in organisations' computers (Huber, 1991).

2.3.1.3.2. The three pillars of knowledge management (Wiig, 1993)

Wiig's (1993) framework is based on three pillars that are supported by a conceptual knowledge management base. The base has four basic knowledge management processes. The first is knowledge creation, which is how knowledge is generated from existing and new knowledge. Another process is knowledge manifestation. It is how knowledge is manifested in peoples' minds, procedures, culture and technology. The third is knowledge use, which entails using knowledge to reason, solve problems, and make decisions. The last pillar is knowledge transfer, which entails learning and capturing knowledge and exchanging it. The three pillars represent other processes. The first pillar has three knowledge processes: analyse knowledge and related activities, elicit, codify and organise knowledge, survey and categorise knowledge. The second pillar has only one process, which is appraise and evaluate knowledge and related actions. The third pillar has three processes; distribute and automate knowledge, handle, use and control knowledge, synthesise knowledge and related activities, and leverage knowledge (Wiig 1993, p. 20). Wiig's framework indicates that knowledge management is not a single process but a series of many processes that are combined to achieve a common goal.

2.3.1.3.3. The design of information products (Meyer & Zack, 1996)

Meyer and Zack (1996) proposed an information processing platform for the manufacture of information products. The platform is based on two notions, i.e. a platform as a repository that contains information content and its structure and a platform as a refinery for information processing. The repository stores all kinds of content to be used by the company. Meyer and Zack's (1996, n.p.) framework describes the architecture of the repository. But their conclusion is that the repository is the foundation on which the firm creates its families of information products. They further state that "the manufacture of information products resembles a refinery". The information processing refinery process is based on five information processing stages: information acquisition, refinement, storage/retrieval, distribution, and presentation or use.

Information acquisition deals with issues of data sources such as its credibility, its accuracy, and its timeliness. Refinement means cleaning and standardising the data. It includes processes such as meta-analysis on information repositories. The main aim of refinement is to add value to the information. Storage and retrieval means keeping the information for future use using computerised databases and knowledge management software. Distribution is making the information available to the end user. The medium, frequency and timeliness are important issues to be considered when distributing information. The final stage is presentation, which entails providing the information to those who need it. Mayer and Zack (1996) state that these processes are not executed sequentially, as there maybe feedback loops between them.

Nickols (1996) also developed a framework that consists of similar processes to those of Meyer and Zack's theory. Nickols' theory has the following processes: knowledge acquisition, organisation, specialisation, store/access, retrieve, distribution, conservation, and disposal.

2.3.1.3.4. Information systems solutions to knowledge management (Skyrme, 1998)

Skyrme's (1998) framework describes technology tools that could be used to support different knowledge management functions. The framework indicates that these tools can support knowledge identification, creation, collection/codification, storage (knowledge database), and diffusion/use. The knowledge identification process can be supported by knowledge discovery and data and text mining tools. Knowledge creation can be supported by thinking aids and conceptual mapping tools. The collection and codification of knowledge can be supported by intelligent agents. Knowledge can be stored in knowledge databases and it could be applied by using video conferencing, groupware and decision support tools.

2.3.1.3.5. The knowledge management process framework (Bukowitz & Williams, 1999)

Bukowitz and Williams' (1999) framework has two broad knowledge management processes: the tactical and the strategic processes. The tactical process is triggered by market-driven opportunity or demand. The strategic process is triggered by shifts in the macro-environment. The tactical side spans four basic steps; people get the information they need for their daily tasks from different sources. They use it to create value, learn from the created knowledge and contribute the knowledge back to the system for others to make use of it. The strategic process spans three steps: assess information – this means organisations have to define their mission and map current intellectual capital against future needs. The other process is to build and sustain an information database. This ensures that knowledge will keep the organisation viable and competitive in future. Organisations also divest the information, which means making it available internally and externally. According to Bukowitz and Williams (1999, p. 8) these processes ensure that organisations use their knowledge to respond to demands and opportunities from the market place.

2.3.1.3.6. Knowledge management systems (Alavi & Leidner, 2001)

Alavi and Leidner's (2001) framework views the knowledge management lifecycle from an information systems perspective. It reviews the roles that are played by information systems in knowledge management. The framework posits that information systems can play four knowledge management roles: knowledge creation, storage and retrieval, transfer, and application. Citing Pentland (1995) and Nonaka and Takeuchi (1995), Alavi and Leidner (2001, p. 116) define knowledge creation as the development of new knowledge and replacing existing knowledge. They state that knowledge is created through social, collaborative and individual processes. They discuss a number of technologies that could be used for knowledge creation. These include groupware, email systems and many others. They further state that organisations create knowledge and learn, but they also have a tendency to forget. To stop forgetting, they create organisational memory to store and later retrieve their knowledge. Their assertion is that in organisations, knowledge is stored in written documents, electronic databases, and peoples' heads (p. 118). They state that computer technology and sophisticated query languages and database management systems could be used to store and retrieve knowledge in organisations.

Alavi and Leidner (2001, p. 119) state that knowledge transfer can take place at various levels in the organisation, at individual, group, and organisational levels. It can also take place across different platforms formally and informally. What is important is that it should be transferred to where it is needed. They state that IT such as video conferencing and lotus notes could be used to transfer knowledge. The application of knowledge in organisational routines is the source of competitive advantage. They further state that IT can play a role in knowledge application by enhancing the integration and application of knowledge. IT facilitates updating, accessibility and capture of knowledge in organisations (p. 122). Like Skyrme's framework, it describes the systems that can be used to support the different knowledge management functions.

2.3.1.3.7. First and second generation knowledge management (McElroy, 2003)

McElroy's (2003) presents a knowledge management lifecycle framework that combines first and second generation knowledge management. According to McElroy (2003, p. 4) first generation knowledge management focused more on technology. Second generation knowledge management focuses more on people, processes and social initiatives. McElroy further explains that first generation knowledge management assumes that knowledge management begins after knowledge production, yet second generation knowledge management emphasises knowledge production and knowledge integration. The framework is based on the second generation notion. The framework is divided into two broad categories of activities: knowledge production and knowledge integration. These broad categories are further divided into many sub-categories. Knowledge production activities involve individual and group learning, knowledge claim formulation, and information acquisition. Knowledge integration activities involve knowledge broadcasting, searching, teaching and sharing.

2.3.1.3.8. The identification and transfer of internal best practices (O'Dell, Grayson & Essaides, 2003)

O'Dell, Grayson and Essaides' (2003) knowledge transfer framework has seven knowledge management processes. They are organising, sharing, adapting, using, creating, defining, and collecting. Although these authors call it a knowledge transfer framework, its processes are similar to the widely used general knowledge management processes.

2.3.1.3.9. The knowledge planning, creation, integrating, organising, transferring, maintaining, and assessing framework (Rollet, 2003)

Rollet's (2003) framework of knowledge management has seven knowledge management processes: knowledge planning, creation, integrating, organising, transferring, maintaining, and assessing. According to Rollet (2003, p. 35) the aim of planning in knowledge management is to break down knowledge management projects into manageable pieces without losing the overall knowledge management goal. In fact, planning involves setting the knowledge management goals. Knowledge creation is defined as the creation of genuinely

new knowledge in the organisation. This could be done by research and development departments, quality assurance departments or anywhere in the organisation (p. 45). Integrating knowledge is the process of making existing knowledge available in the organisation. This could be external or internal knowledge. It could be done in a number of ways, for example, staff training, cooperation with other organisations and buying knowledge products. Organising knowledge could be defined as the process of arranging knowledge according to its characteristics so that it is easy to access. Rollets (2003, p. 69) states that knowledge is organised to increase effectiveness and efficient retrieval when needed. Knowledge has to be transferred in order to be used (Rollet, 2003). Knowledge could be transferred using pull and push mechanisms including face-to-face interactions. Knowledge assessment serves to determine whether or not goals set out at the planning stage have been met.

2.3.1.3.10. Knowledge management processes (Beccerra-Fernandez, Gonzalez & Shaberwal, 2004)

Beccerra-Fernandez *et al.'s* (2004) framework has four main knowledge management processes and seven sub-processes. According to Beccerra-Fernandez *et al.* (2004, p. 33) these processes and sub-processes are based on Nonaka (1994), Grant (1996a, 1996b) and Nahapiet and Ghoshal (1998) theories. The four main processes are knowledge discovery (which consists of two sub-processes, socialisation and combination), knowledge capture (which consists of externalisation and internalisation sub-processes), knowledge sharing (which consists of socialisation and exchange sub-processes), and knowledge application (which consists of direction and routines sub-processes). According to Beccerra-Fernandez *et al.* (2004, p. 33) knowledge discovery is the generation of new knowledge, tacit or explicit, from information or synthesising prior knowledge. They further state that new explicit knowledge is discovered by the combination of new tacit knowledge through socialisation. They define knowledge capture as the process of retrieving tacit or explicit knowledge from the organisation's knowledge sources. These sources could be people, artefacts, or

organisational databases. This is done through externalisation and internalisation (p. 33-34). They define knowledge sharing as the process through which knowledge is communicated to other people through socialisation. Knowledge is applied to guide decisions and actions. It is applied in organisational routines and processes.

Also in 2004, Awad and Ghaziri (2004) developed a framework with four knowledge management processes: knowledge capturing, organising, refining, and knowledge transfer.

2.3.1.3.11. The knowledge management lifecycle (Dalkir, 2011)

Dalkir's (2011) integrated framework was created based on Wiig (1993), Meyer and Zack (1996), McElroy (1999), Bukowitz and William (1999), and Rollet (2003) frameworks. Dilkir combined the processes found in the frameworks developed by the above mentioned scholars to create an integrated framework. The framework has three main stages: knowledge sharing and dissemination, knowledge capture and/or creation, and knowledge acquisition and application. Dalkir (2011, p. 53) defines knowledge capture as the identification and codification of existing internal and external knowledge. Knowledge creation is the development of new knowledge. This could represent innovations such as new products and processes. Knowledge transfer and dissemination is defined as the process of making the knowledge available to those who need it. Knowledge is then applied to organisational routines and processes. The framework shows that the transition of knowledge across these three stages happens through assessment which is a criterion that will inform organisational goals. Contextualisation involves maintaining the link between the holders of knowledge and the knowledge, and updating. That means validating the usefulness of knowledge and keeping it up-to-date.

2.3.1.3.12. Knowledge creation, sharing, structuring, using and auditing framework (Sağsan, 2006)

Sağsan's (2006) framework has five processes: creating, sharing, structuring, using and auditing knowledge. According to Sağsan and Zorlu (2010:405), knowledge is generated by converting tacit and explicit knowledge. They further state that it is shared through social and technological channels. Knowledge auditing allows organisations to control their intellectual capital. Sağsan (2006) states that organisations use knowledge to gain competitive advantage, design and market products, and improve the organisation's services quality.

2.3.1.3.13. The identify, organise, store, apply, evaluate, and create (IOSAEC) framework (Evans & Ali, 2013)

The identify, organise, store, apply, evaluate, and create (IOSAEC) knowledge management lifecycle framework was developed by Evans and Ali (2013). It consists of six knowledge management processes which are identify, organise and store, share, apply, evaluate and learn, and create knowledge. They define knowledge identification as formally identifying the knowledge assets before attempting to manage them. This helps the organisation to put boundaries around tangible and intangible knowledge assets (p. 161). Knowledge organisation comprises the sorting, classification and categorisation of stored knowledge. Sharing knowledge is making it available to those who need it within the organisation. Knowledge could be shared internally and/or externally.

This framework is built from other knowledge management frameworks. The application of knowledge happens when employees act on information and artifacts (Evans & Ali, 2013, p. 162). This stage allows workers to adopt new approaches and be innovative. The results of the knowledge application are learning in the organisation. This is because, by using knowledge, organisations acquire new knowledge and learn from previous experiences. Learning improves personal experience and improves performance. Evans and Ali (2013) are of the opinion that employees create new knowledge from the learning activities they go

through. That is, as they do things better, they develop new knowledge that could be used in future activities.

A synthesis of the 15 frameworks briefly discussed above reveals many similarities. It could be seen that these frameworks explain how organisations manage knowledge. Although there are quite a number of different processes discussed in the frameworks, six processes stand out. Organisations identify/acquire and create knowledge, they store and organise it for future use in organisational databases, they share it with those who need it and people use or apply the knowledge in their daily tasks. The only difference that is noticeable is the use of terminology and the sequence of the processes. For example, some frameworks use knowledge dissemination while others use knowledge sharing and transfer interchangeably. The sequencing of the processes is also not uniform. For example, one framework could start with knowledge creation while in another knowledge creation is the last step. Other scholars have indicated that the sequencing does not matter because feed-back loops could apply.

For the purposes of this study, no single framework was chosen to inform the study. Instead, a new framework was developed, which was inspired by the fact that existing frameworks do not seem to adopt one single terminology for the knowledge management processes. The new framework will address that. There is a lot of duplication of processes in existing frameworks. The processes are almost similar in all the frameworks. The new framework will combine all the similar processes and come up with one group of knowledge management processes.

The following section will explain the methodology used to build the new framework and lastly present the new framework informing the study.

2.4. Building the knowledge management conceptual and theoretical framework

This section describes how the knowledge management lifecycle conceptual and theoretical framework was created from the 15 frameworks briefly discussed. The following criterion was used to develop the framework.

Prominent processes were counted, and the number of times they appeared in all the frameworks was determined. If a process appeared in more than five frameworks, it was accepted as a prominent process; therefore, it was included in the new framework. In some instances, terminology was taken into consideration. It was noticed that there are similar processes using different terms, for example, terms such as knowledge transfer/distribution/dissemination/sharing. In such cases, they were grouped into one process; for example, 'knowledge sharing'. The following processes were grouped together as one: knowledge acquisition/discovery/identify/sourcing/get/collect. They were grouped into the 'knowledge acquisition' process. Knowledge transfer/distribution/dissemination/sharing/contribute terms were grouped into the 'knowledge sharing' process. Knowledge application/use terms were grouped into the 'knowledge application' process. Knowledge storage/conservation and organisational memory/databases were grouped into the 'knowledge storage' process. This process also includes terms such as knowledge storage and retrieval and knowledge storage and access.

The following processes were not included in the conceptual and theoretical framework because they did not meet the criterion of being prominent: knowledge identification, auditing, discovery, planning, assessing, defining, adapting, maintaining, building, divest, specialisation, disposal, presentation, compilation, transformation, value realisation, and interpretation. They appeared in fewer than five frameworks. Although they have not been

included in the framework, it does not mean that they are not important. They could be integrated into the main processes. Table 4 shows the concepts. Column one shows the prominent concepts that were selected for inclusion in the conceptual framework. Column two shows the frameworks they were taken from.

Table 4: Knowledge management processes of the proposed framework

KM processes	**Framework**
Knowledge acquisition	Huber (1991); Wiig (1993); Meyer and Zack (1996); Nickols (1996); McElroy (2003); O'Dell, Grayson and Essaides (2003); Dalkir (2011).
Knowledge sharing	Huber (1991); Wiig (1993); Nickols (1996); Bukowitz and Williams (1999); Alavi and Leidner (2001); Rollet (2003); O'Dell *et al.* (2003) Becerra-Fernandez, Gonzalez and Sabherwal (2004); Dalkir (2011); Sağsan (2006,2007, 2009); Evans and Ali (2013).
Knowledge creation	Wiig (1993); Skyrme (1998); Alavi and Leidner (2001); O'Dell *et al.* (2003); Rollet (2003); Dalkir (2011); Sağsan (2006, 2007, 2009).
Knowledge application	Wiig (1993); Skyrme (1998); Alavi and Leidner (2001); O'Dell *et al.* (2003); Becerra-Fernandez *et al.* (2004); Dalkir (2011); Sağsan (2006, 2007, 2009), Evans and Ali (2013).
Knowledge storage	Huber (1991); Meyer and Zack (1996); Nickols (1996); Skyrme (1998); Alavi and Leidner (2001); Evans and Ali (2013).
Knowledge organisation	Nickols (1996); Rollet (2003); Awad and Ghaziri (2004); Sağsan (2006, 2007, 2009); Evans and Ali (2013).

2.5. Concepts of the knowledge management framework

This section discusses the concepts of the new knowledge management lifecycle framework informing the study. Existing frameworks do not show any particular order in which the processes appear. This framework also does not show any particular order which the processes follow. It subscribes to the notion that feedback loops might occur. For example, knowledge could be acquired from an external source and used immediately before it is refined or organised. Generally, however, the literature reveals that organisations experience knowledge gaps which they try to fill by acquiring knowledge from internal and external

sources. The acquired knowledge is used to create new knowledge within the organisation. Once the knowledge is created, it is then refined, organised and stored in knowledge databases for future use. Knowledge stored in knowledge databases is then shared and used within the organisation using different channels. While the knowledge is used, individuals, groups and the whole organisation use and learn from the knowledge. In no particular order, the processes of the unified lifecycle framework are discussed below.

2.5.1 Knowledge acquisition

According to Holsapple and Joshi (2002) knowledge acquisition is getting knowledge from the external sources and turning it into a commodity that can be used within the organisation. They state that knowledge acquisition includes extracting knowledge from external sources such as locating, accessing, capturing, and collecting knowledge from customers, competitors, suppliers and other knowledge sources. It also includes extracting the knowledge, that is, transforming it and packaging it into usable knowledge. This process also includes interpreting the knowledge so that it can be transferred easily to people or activities that need it (Holsapple & Joshi, 2002, p.56). Knowledge acquisition is similar to what Bukowitz and Williams (1999, p.36) call knowledge 'get', which means getting the knowledge from a particular source. Gendreau and Robillard (2013) discuss knowledge acquisition activities in software development. They state that knowledge is acquired from internal and external sources in a software development environment. Examples of external sources are the Web, technical documentation, a book or paper. Internal knowledge is the tacit knowledge in the developers' heads (p. 2). Zheng (2012, p. 2667) states that knowledge acquisition refers to getting knowledge and experience from different sources. Zheng further states that the acquired knowledge should be incorporated into procedures.

2.5.2. Knowledge creation

The concept knowledge creation was made popular by Nonaka and Takeuchi (1995) in their famous organisational creation theory in which they indicate that knowledge is created through the conversion of tacit and explicit knowledge in four modes: socialisation, externalisation, combinations and internalisation. Knowledge creation involves developing new knowledge and replacing existing knowledge with the new knowledge (Pentland, 1995, p. 3). Mitchell and Boyle (2010, p. 69) define knowledge creation as processes or activities, as an output of those processes and activities, or outcomes that add value. For example, products and services. Knowledge creation is a process when initiatives and activities are undertaken towards the generation of new ideas or objects. As an output, knowledge creation is the development of new ideas add value to existing knowing. Knowledge is created in five steps: sharing tacit knowledge, creating concepts, justifying concepts, building a prototype, and cross levelling knowledge (Wan *et al.*, 2010, p. 488). Gottschalk (2007, p.31) states that knowledge is created through a social collaborative process and individuals' cognitive processes. In software development, knowledge creation takes two forms, i.e. creating the output (software) or the process of creating the output (systems development lifecycle (SDLC) processes).

2.5.3 Knowledge organising

According to Pentland (1995, p.3) knowledge organisation is the grouping and integration of knowledge bodies. Awad and Ghaziri (2004, p.24) define knowledge organisation as the cataloguing, indexing, filtering, and the codification of knowledge. This enables knowledge to be easily retrieved. Rollet (2003, p.12) states that organising knowledge involves putting it in context and structuring the knowledge through hierarchical classification or knowledge mapping. Chouseinoglou *et al.* (2013) defines it as the association of knowledge with each

other, leading to an organised knowledge structure. Samoilenko and Nahar (2013, p.1356) are of the opinion that knowledge is organised to allow users to have easy access to it. They further state that knowledge is organised through indexing, abstracting and packaging and cataloguing. In the software engineering field, knowledge organisation is organising software development knowledge stored in organisational databases for easy retrieval. Hodge (2000, p.1) gives a broad Library and Information Science definition of knowledge organisation. According to Hodge, knowledge organisation includes classification and categorisation schemes that organise material.

2.5.4 Knowledge storage

Samoilenko and Nahar (2013, p. 1355) define knowledge storage in the software engineering perspective. They define it as the process of gathering the relevant knowledge and keeping it for use in complex systems development tasks and achieving organisational goals. They further state that knowledge can be stored in various formats such as the Internet, printed paper forms (manuals, books, documents, journal articles, and other sources), and organisational databases (p. 1353). In order for organisations not to forget or lose knowledge, they store their knowledge for future use, both in electronic databases and human heads. Such databases are called organisational memory.

According to Stein and Zwass (1995, p.85) organisational memory is the means by which knowledge from the past influences present organisational activities. Andrade *et al.* (2013, p.13) state that an organisational memory specifies and supports the representation and dissemination of important organisational knowledge. Basili, Caldiera and Rombach (1994) call them experience factories. “The experience factory recognises that organisations need to learn from their past experiences in order to” deliver high quality products quicker, and cheaper (Basili, Caldiera & Rombach 1994, n.p.). The experience factory explains how

software development projects can store and use knowledge from past projects to prevent 're-inventing the wheel'. It is a database of lessons learned and best practices. Dingsoyr (2000, p.2) describes the experience factory as infrastructure for reusing lifecycle experiences, processes and products for software development. Dingsoyr (2000) further states that experiences are collected from software development projects, packaged and stored in experience databases. Basili, Caldiera and Rombach (1994, p.8) are of the view that improving software development processes and products requires the capture and storage of experiences into databases that are accessible and updated to meet the needs of current projects. It packages the experience by building formal and informal models of various software processes and products and other forms of knowledge via documents, automated support and people.

Examples are product packages which are about the life cycle of products. Another example is data packages which is data relevant for software project purposes and tools packages which are instructions for the use of a tool and experience with it. Dingsoyr (2000, p.1) states that the experience factory is a knowledge management process that can help create a learning environment among software developers. These sentiments are shared by Basili *et al.* (1994, p.16) who state that the experience factory facilitates learning and reuse of knowledge and creates tangible corporate asset in the form of packaged experiences.

The experience factor framework supports two knowledge management processes: knowledge storage and application. This is because it promotes the storage of experiences and their reuse in the software development process. An example of an experience factory or organisational memory is the National Aeronautics and Space Administration (NASA) Systems Engineering Body of Knowledge (SEBoK), a database of all software development code generated by the agency (Jansma & Means, 2012).

2.5.6 Knowledge sharing

The concepts knowledge sharing and transfer are sometimes confused and used interchangeably. According to Major and Cordey–Hayes (2000, p. 412), knowledge transfer is conveying knowledge from one place or person to another. Li and Gao (2003) give a business definition of knowledge transfer as associated with emulating and leaning continuously from competitors. Becerra-Fernandez, Gonzalez and Sabherwal (2004, p. 34) define knowledge sharing as the process in which knowledge is communicated to other people. This process can take place across individuals, groups, and departments within the organisation. The knowledge shared must be effective so that the recipient can understand it. Razak *et al*. (2016, p.547) states that it is a process of getting experience from others. They further state that it is the practice of exchanging and disseminating ideas with others to ensure that the knowledge is sustained and used for the development of the organisation.

Witherspoon, Jason, Cam, & Dan (2013, p. 252) define knowledge sharing as the intention or expectation to exchange information, skills, or expertise with others for the benefit of the sharer." Amayah (2013) states that knowledge is shared face-to-face, by using ICTs such as telephones and email, and that it can be shared informally and formally.

In software development, knowledge sharing involves knowledge sharing among individuals and teams, especially if the teams are physically dispersed (Mathrani, Parsons & Mathrani 2012).

2.5.7 Knowledge application

Knowledge application is actually using the knowledge that has been captured and stored in organisational databases or knowledge in people's heads. Holsapple and Joshi (2002, p. 57) state that it is utilising available knowledge to create new knowledge and produce extra knowledge. Knowledge has to be applied in organisational routines and processes according to Gottschalk (2007, p. 37). The knowledge-based view of the firm indicates that the source of competitive advantage lies in the application of knowledge (Alavi & Leidner, 2001, p.122). These sentiments are shared by Gottschalk (2007) who emphasises the important of applying knowledge in organisational processes and routines.

Nesheim, Olsen and Tobiassen (2011, p. 837) state that knowledge application happens when the experiences of individuals or units influence the change in behaviour of other individuals or units. According to Song, Van der Bij, Weggeman (2005, p. 435) knowledge application is a timely response from the organisation to technological change by applying the knowledge and technology created into new products and processes. Rastogi (as cited in Ranjbarfard, Aghdasi, López-Sáez, López, 2014, p.498) state that knowledge application is retrieving and using knowledge in supporting decisions, organisational routines, training, problem-solving and actions.

The six concepts form the study's knowledge management conceptual framework. There is no fixed order in which these concepts follow each other. Figure 3 shows the lifecycle.

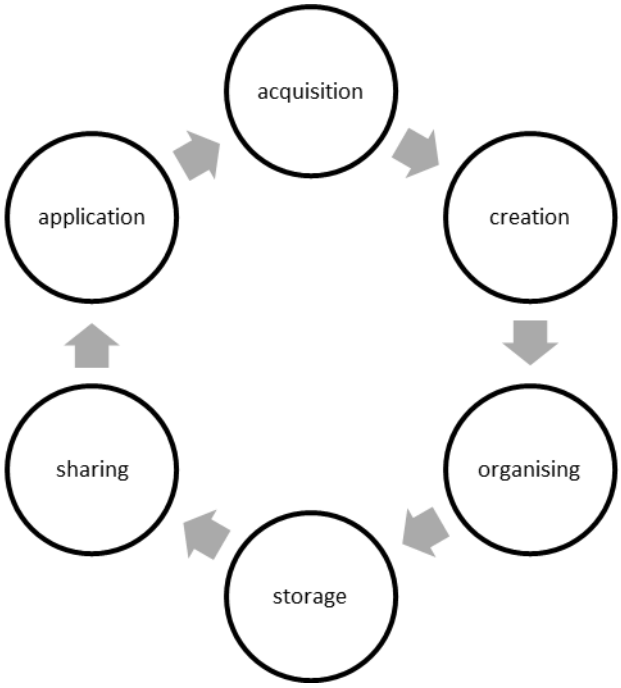


Figure 3: knowledge management lifecycle

2.6. Other relevant theories and frameworks

Apart from the framework developed specifically for this study, there are other knowledge management theories and frameworks that could be used to inform the study. This section briefly presents them.

2.6.1. The theory of organisational knowledge creation (Nonaka & Takeuchi, 1995)

The theory describes how individuals, teams and entire organisations generate new knowledge. The core of the theory lies in the mobilisation in four modes of knowledge conversion that are created when tacit and explicit knowledge interact with each other. The theory assumes that knowledge creation lies in the mobilisation and conversion of tacit and explicit knowledge in four modes: socialisation, externalisation, combination, and internalisation (SECI). These four modes constitute the engine of the entire knowledge creation process. The conversion of tacit to tacit knowledge is called socialisation. When tacit knowledge is converted to explicit knowledge that process is called externalisation. The conversion of explicit knowledge to another explicit form is combination. And internalisation

is converting explicit knowledge to tacit knowledge (Nonaka & Takeuchi 1995; Nonaka, & Konno 1998).

The ontological assumption is concerned with the levels of knowledge creating entities, at individual, group, organisational, and inter-organisational levels. The theory assumes that knowledge is created first at individual level, then at group, organisational, and inter-organisational levels (Nonaka & Takeuchi 1995).

The theory explains five knowledge creation enabling conditions: intention, autonomy, fluctuation and creative chaos, redundancy and requisite variety (Nonaka 1994; Nonaka & Takeuchi 1995).

The theory also presents five methods of the knowledge creation process in organisations: sharing tacit knowledge, creating concepts, justifying concepts, building archetypes, and cross levelling of knowledge. During the tacit knowledge sharing stage, individuals share emotions, feelings, and mental models through face-to-face interactions. Creating concepts involves the sharing of both tacit and explicit knowledge. After sharing tacit knowledge, organisational members then articulate the tacit knowledge into explicit knowledge through written concepts. Justifying concepts is a phase in which created concepts are confirmed as to whether or not they are useful in the organisation. The justified concepts are then converted into tangible or concrete products or a model, called an archetype. After knowledge has been created, it is then levelled across the organisation. This could be done inter-organisationally or intra-organisationally (Nonaka 1994; Nonaka & Takeuchi 1995).

The theory was extended to include the concept of *'ba'*, a shared context where knowledge is created. *'Ba'* could be physical place such as an office or a virtual space. Four types of *'ba'* are given: originating, dialoguing, systematising, and exercising '*ba'* (Nonaka, & Konno 1998:16).

This theory has been used to inform knowledge creation processes in software development environments. Arent and Norbjerg (2000) used the theory to study SPI projects from the knowledge creation perspective. They focused on knowledge creation activities based on the interplay between tacit and explicit knowledge and the four knowledge creation processes (socialisation, externalisation, combination and internalisation.). Among others are Linden and Cybulski (2009), and Wan *et al.* (2010). These studies investigated knowledge creation processes in software development in the context of the SECI process.

On the other hand, the SECI model has been criticised by Gourlay (2006), Poell and Van de Krogt (2003), among others. These authors state that the theory is flawed because of a lack of empirical rigour, the omission of tacit knowledge and the subjective definition of knowledge. They also dispute Nonaka's notion that workers learn within the boundaries set by management and its misunderstanding of Polanyi. Nonetheless, this theory is hailed as one of the best knowledge creation theories.

2.6.2. Communities of practice theory (Wenger, 1998)

Communities of practice present a theory of learning that starts with an assumption that people learn and become who they are because of the social practice engagements (Wenger 1998). Wenger states that the theory focuses on learning as social participation (p.4). Citing Lave and Wenger, Smith (2009, n.p.) states that communities of practice are groups who share similar interests and characteristics for something they do and learn how to do it better as they regularly interact. Smith further states that the characteristics of such communities vary; some are formal, some informal, some have names while others don't. Wenger (2009, n.p.) states that communities of practice are formed by people engaged in a collective learning process in a shared domain of human endeavour. They learn together how to better fulfil passion or address their concerns. Three characteristics of communities of practice are given by Wenger (1998): the domain, the community and the practice. Wenger (1998, p.47)

defines a practice as acting in a social and historical context that gives structure and meaning to what people do. It involves language, tools, documents, images, symbols, procedures, and other various practices. A community is a collection of people. To summarise, the theory states that in organisations, learning can take place through communities of practice. Communities of practice are informal (sometimes formal) groups of people who share common interests. In order for a community of practice to exist, it must have a domain (the interest that is shared), a practice (an activity that they perform) and they must be a community (a group with a shared interest, for example, knowledge managers or software developers). The primary assumption of this theory is that human knowing is a social act.

This theory has been also used extensively to study software development phenomena. Mestad *et al.* (2007) used it to study learning communities in a small software organisation. Paasivaara and Lassenius (2014) used the same theory to investigate communities of practice in distributed agile teams where they discovered that such teams promoted knowledge sharing and learning, technical work and coordination.

2.6.3 The experience factory framework

This is a knowledge management framework developed by Basili *et al.* (1994) to help software organisations reuse knowledge to improve software quality and reduce development costs. According to Basili *et al.* (1994, p.1) the 'experience factory [is] aimed at the capitalisation and reuse of lifecycle experience and products'. Dingsoyr (2000, p. 2) describes the experience factory as an infrastructure for reusing lifecycle experiences, processes and products for software development. Dingsoyr further states that experiences are collected from software development projects, packaged and stored in experience bases. Dingsoyr (2000, p.1) states that the experience factory is a knowledge management process that can help create a learning environment among software developers. The experience factor framework supports two knowledge management processes: knowledge storage and

application. This framework promotes the storage of experiences and their reuse in the software development process.

2.6.4. The learning organisation (Senge, 1990)

This theory was developed by Senge in the early 1990s. The aim of this theory was to portray a new kind of organisation shaped by the learning of its members, thus making it a learning organisation. According to Senge (1990, p.13) a learning organisation is "a place where people are continually discovering how they create their reality and how they change it". Senge further states that in such organisations, people keep on improving their ability to create their truly desired results. According to Senge (1990), these organisations are able to learn faster than their competitors. Senge (1990) gives five core disciplines or characteristics of the learning organisation, which are: systems thinking, personal mastery, mental models, building a shared vision and team learning. The disciplines are briefly discussed below.

Systems thinking

According to Senge (1990, p.7) systems thinking is a "conceptual framework, a body of knowledge and tools that have been developed to make full patterns clearer, and to help us see how to change them effectively". Systems thinking enables individuals and organisations to see things as part of a whole (system) instead of seeing them as bits and pieces disjointed from the whole.

Personal mastery

Senge (1990, p. 141) states that personal mastery is not only competence and skills, but it is grounded in competence and skills. He states that it is a discipline of personal growth and learning, learning how to generate and sustain creative tension in peoples' lives. Creative tension is a force that brings people together to seek resolutions to problems. According to Senge (1990, p. 142) people with a high level of personal mastery share several basic

characteristics such as living in a continual learning mode, never arriving; they are aware of their ignorance, their incompetence, and their growth areas. These characteristics allow them to continuously learn and create learning organisations.

Mental models

Senge (1990, p. 8) defines mental models as assumptions, generalisations, pictures or images that influence how people understand the world and how people take action. This discipline deals with how the organisation is viewed. It is the ability to carry on learning conversations that balance inquiry and advocacy and where people are free to expose their thinking effectively and for that thinking to influence others. Mental models can be simple or complex theories. Their main objective is to allow people to learn through assumptions, generalisations and pictures. Mental models are also emphasised by Nonaka and Takeuchi (1995) in their model of organisational knowledge creation.

Shared vision

Senge (1990, p. 206) describes a shared vision as a force in peoples' hearts, a force of impressive power. He further states that a shared vision provides the focus and energy for learning. It uplifts people's aspirations, making work part of the larger purpose embodied in the organisation's products and services, thus accelerating learning. A shared vision creates a spark, which is the excitement that lifts the organisation out of the mundane. It changes employees' relationships with the organisation, making the company theirs, and they start working together to achieve common organisational goals (p. 208). Senge moreover observes that there will be no learning organisation without a shared vision. This is because a shared vision facilitates learning when challenges occur.

Team learning

Team learning is the process of aligning and developing the capacity of a team to create the results its members truly desire (Senge, 1990, p. 236). Team learning builds on personal mystery. It is a microcosm for learning throughout the organisation. Team learning has three critical dimensions: the need to think about complex issues, the need for innovative, coordinated action, and the role of the team members on other teams. Learning teams continually foster other learning teams by inculcating the practices and skills of team learning broadly (p. 236-237).

2.7. Software development and software development failures

The second part of the conceptual and theoretical framework discusses concepts from the software engineering field. Specifically, it conceptualises software development and software development failure, its types and causes. It is important to have a clear understanding of what software development and software development failures are because of the many interpretations given to these concepts.

2.7.1. Systems/software development

Stair and Reynolds (2012, p. 28) define systems development as a process or activity of creating or modifying information systems. There are a number of software development methods that have been developed over the years. Tilley and Rosenblatt (2017, p. 17) state that the most popular software development methods are structured analysis, object-oriented analysis, and agile methods. They further state that other organisations choose to use CASE tools. Structured analysis is a series of phases called the systems development life cycle (SDLC). The SDLC has the following phases: planning, analysis, design, implementation, support and information system security (Tilley & Rosenblatt, 2017, p. 18). An example of structured analysis is the waterfall method.

2.7.2. SDLC and the waterfall method

There are many versions of the SDLC and the waterfall software development method, but basically it has five major stages of development: systems planning, analysis, design, implementation, security and support (Tilley & Rosenblatt, 2017, p. 20). Cobb (2011, p. 6) states that the waterfall method is a series of processes that sequentially follow each other and has five main processes: requirements, design, development, integration and testing, and implementation and development. Stair and Reynolds (2012, p. 29) state that the SDLC has the following processes: systems investigation, analyses, design, implementation, maintenance and review. Satzinger, Jackson and Burd (2014, p. 6) state that there are six core processes in the SDLC: identification of the problem or need, planning and monitoring, discovery and understanding of the project details, designing the system, building, testing and integrating the system components, and complete systems tests and deploying the solution. What is deduced from the literature is that the SDLC is broad and has different phases. We assume that it depends on the development organisation which phases it will choose to adopt for development purposes. For the purposes of this study the Tilley & Rosenblatt (2017) SDLC model was adopted. It has most of the phases given by other authors. The waterfall method will be briefly discussed in the following subsection.

2.7.2.1 Processes of the waterfall method (Tilley & Rosenblatt, 2017)

According to Tilley and Rosenblatt (2017, p. 20) the waterfall method has the following phases: systems planning, analysis, design, implementation, security and support. Each phase is briefly discussed.

Systems planning

This process involves determining system requests, which are the desired changes to an information system. The main purpose of this process is to perform preliminary

investigations to evaluate an IT related business opportunity. It also involves organising and scheduling the project (Tilley & Rosenblatt, 2017, p.20; Satzinger, Jackson & Burd, 2014, p. 210). Alwan (2015, n.p.) asserts that at this stage the problem is defined, objectives set, and resources allocated. It also involves studying the ability to propose alternative solutions and studying to differentiate your product from those of competitors.

Systems analyses

According to Tilley & Rosenblatt (2017, p. 20) this process aims to build a logical model for the new system. This involves requirements modeling where the systems analyst investigates business documents and processes to determine what the new systems must do. Alwan (2015, n.p.) notes that this stage involves determining the user requirements, documenting them and conducting a feasibility study for the project.

Systems design

Systems design involves the creation of a physical model that will satisfy all documented requirements for the system. User interfaces are designed at this stage, inputs, outputs and processes are clearly spelled out and internal and external controls are designed (Tilley & Rosenblatt, 2017, p. 20). It involves configuring and structuring the new system (Satzinger, Jackson & Burd, 2014, p. 210).

Systems implementation

This is the phase in which programs are written, tested, and documented and the system installed. System analysts configure the software and perform necessary modifications. The main objective of this phase is to deliver a complete and documented working system (Tilley & Rosenblatt, 2017, p. 21). Satzinger, Jackson and Burd (2014, p. 210) explain that this phase includes programming and testing the system.

Systems security and support

This phase entails system maintenance, enhancements, and security. It deals with bugs, provide new features if necessary, and internal and external security controls (Tilley & Rosenblatt, 2017, p. 21). Figure 4 shows the waterfall model.

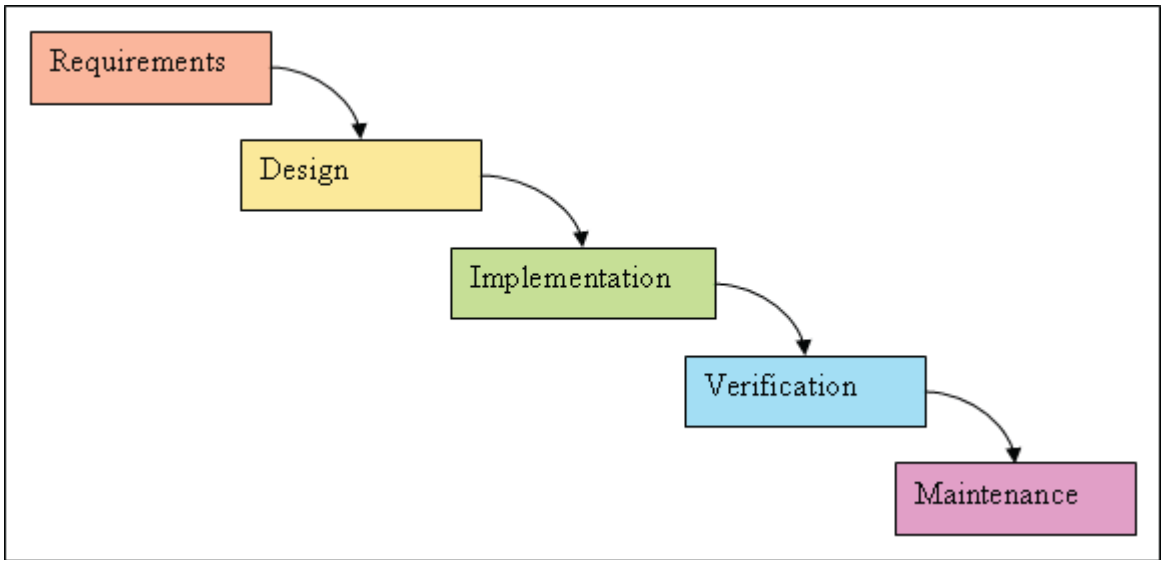


Figure 4: The waterfall method (Source: Serena, 2007, p.4)

Moniruzzaman and Hossain (2013) note that the advantages of traditional methods of systems development are that they provide predictability and high assurance, conform to plans, have heavy detailed explicit knowledge and that they are clearly defined and well documented.

2.7.2.2. Agile methods

An alternative to traditional (waterfall) methods are agile software development methods. Boehm (2002) states that agile methods have short iterations, frequent releases, peer review, on-site customer participation, and simple design. The Agile Manifesto (2001) asserts that agile methods value individuals and interactions over processes and tools; they prefer working software over comprehensive documentation; focus on customer collaboration over contract negotiation and respond to change over following a plan. The main reason agile

methods were developed was to counter the challenges faced by traditional software methodologies (Cao & Ramesh, 2007, p. 41) which are believed to be responsible for high project failure rates. Cao and Ramesh (2007, p. 41) noted that agile methods prefer "short iterations, frequent releases, simple and emerging design, peer review, and on-site customer participation". Agile methods stress quality design which is an ongoing process throughout the development phase and done in smaller chunks as opposed to all at once and up-front development (Highsmith & Cockburn, 2001, p.120).

According to Serena (2007) there are two main agile software development methodologies. They are Extreme Programming (XP programming) and Scrum. XP concentrates more on the development rather than the managerial aspects of the project. In a Scrum environment teams get together to develop the software with a purpose of putting out a release. It focuses on both development and management of the project (p.6-8). Boehm (2002, p. 66) observes that XP "advocates doing extra work to get rid of architectural features that do not support the system's current version" while Scrum adopts end-of-iteration reviews with customer focus groups (Highsmith & Cockburn, 2001, p.122). Moniruzzaman and Hossain (2013, n.p.) are of the view that XP focuses on project level activities of implementing software and Scrum is aimed at managing the project while increasing the probability of success.

The main difference between the waterfall method and agile methods is in the linear and dynamic structures that these two methodologies take. For example, the waterfall takes a linear structure with different phases to complete. Phases follow each other sequentially. Deliverables are expected at each phase. Agile methods on the other hand prefer iterations rather than phases. This means each stage of the development process could be taken at any given time. The phases do not have to follow each other sequentially. "The output of each iteration is working code that can be used to evaluate and respond to changing and evolving user requirements" (Serena, 2007, p. 5). Moniruzzaman and Hossain (2013, n.p.) provide

more than ten differences between the two methodologies. Among them are that agile methods are dynamic and embrace change quickly while traditional methods have difficulty adapting to changing environments. They further state that agile methods rely mainly on tacit knowledge while traditional methods rely mainly on explicit knowledge.

Agile methods are perceived to have many advantages compared to traditional methods. Communication is one benefit. This is because agile development emphasises continuous communication throughout the development phase (Paasivaara & Lassenius, 2006, n.p.). Short iterations are other benefits as given by Paasivaara and Lassenius (2006). This is because they bring transparency of work progress to all stakeholders. Soundararajan and Arthur (2011, p. 14) give the following advantages of agile methods; the ability to accommodate changes throughout the development lifecycle, better predictability, improved customer satisfaction, improved quality, better team morale, greater return on investment, reduced waste, shorter development periods, and satisfaction. According to Maher, Kourik and Chookittikul (2010, p. 300) "their flexibility provides the means to address many common problems faced in the development of modern software systems such as changing business requirements, staff turnover, delayed deadlines, cancellations, and applications failing to match customer requirements".

2.7.3 Computer Aided Software Engineering (CASE), joint application development (JAD) and rapid application development (RAD)

Apart from adopting specific methodologies, software developers use software development tools that help them develop software. Popular tools are JAD, RAD and CASE tools. JAD and RAD tools use teams composed of IT staff, managers, and users. JAD focuses on team-based fact-finding and RAD is more like a compressed version of the JAD process (Tilley & Rosenblatt, 2017, p. 24). Stair and Reynolds (2012, p. 509) state that RAD tools are a combination of tools, techniques, and methodologies designed to speed up the development

process. CASE tools are a set of individual tools that share a database of information. Examples are database tools, modelling tools, documentation tools and other useful tools (Tilley & Rosenblatt, 2017, p. 486). Stair and Reynolds (2012, p. 515) state that CASE tools automate the tasks in systems development and encourage adherence to the SDLC, thus instilling a high level of rigour and compliance to the whole development process.

2.8. Software development project failures

Software development project failures have been a big concern for software development organisations. This is an issue that has not been addressed for a long time. Ewusi-Mensah (2003, p. 7) says that the issue of software development failure has been present since the early days of the computing revolution and is likely to be with us even in future. Linberg (1999, p. 177) states that software development failures have become commonplace and that these failures are reported almost daily in the media. A failed project could be described as a project that is terminated before its completion (Pinto & Mantel, 1990, p. 261). Linberg (1999, p. 190) states that software project failure is defined in terms of its completion or cancellation. Linberg (1999) cautions though that a project could be completed but still be deemed a failure if it does not meet quality expectations. Lyytinen and Hirschheim (1987, p.263) define failure as the inability of a system to meet a certain stakeholder's expectation. They further state that failure comes in two ways: structural and processual, the former meaning that the information system (IS) designed affects the structural properties of the stakeholder's environment such that it clashes with the stakeholder's interest, while the latter means that the stakeholder's interests have changed due to many reasons. For the purposes of the study, a software project failure is a project that is either stopped before its completion or a project that has been completed but never used because the customer is not satisfied with the system.

2.8.1. Types of software development project failures

The literature reports a number of software project failure types. Lyytinen (1988, p.47) is of the opinion that failure could be a developmental error and by use. The former refers to a situation in which the system is not created to suit the users' interest/need and in the latter case the system is not aligned with the users' interests. The Standish Group Report (Chaos) (1995, p. 1) classifies project failures into challenged and impaired. Challenged projects are completed, operational but over budget and over the time estimate, with fewer features and functions than anticipated. Impaired projects are those that are cancelled at some point during the development cycle. Ewusi-Mensah and Przasnyski (1991) call them abandoned projects.

The Standish Group's CHAOS Manifesto (2013, p.1) classifies projects into successful, challenged and failed. A successful project is "delivered on time, on budget, and with required features and functions." A challenged project is "late, over budget, and/or with less than the required features and functions." A failed project is "cancelled prior to completion or delivered and never used". Ewusi-Mensah (2003, p. 7) calls the latter an abandoned project. He further classifies failures into two classes: a system that is delivered but does not perform according to expectations and a system not completed at all by developers (abandoned). Lyytinen and Hirschheim (1987, p.264) classify failure into four types: correspondence failure, process failure, interaction failure, and expectation failure.

Correspondence failure happens when design objectives are not met in advance. This failure is usually associated with management's failure to spell out the system's objectives well in advance (before the system is designed). Process failure happens when the system has exceeded its budget. This failure is associated with management's inability to allocate enough resources for the project. Interaction failure has to do with the non-acceptance of the system by users. Expectation failure happens when the system's expectations are not explicitly defined or when vague expectations are given. Bupa (2005, p. 16) gives two types of IS

failures: failure in the development of major new systems and in the enhancement to an existing system. According to Bupa (2005) the former is caused by poor project management.

Different researchers have classified different types of project failure, but these failures are not completely different at all from one another. For example, what is referred to by the Standish Group as an impaired project is what is referred to by Ewusi-Mensah and Przasnyski (1991) as an abandoned project. What is referred to by the Standish Group as a challenged project is what is called by Lyytinen and Hirschheim and Ewusi-Mensah an interaction and expectation failure. The study adopted the definition of failure given by The Standish Group. This is because this definition is broad. By definition it covers most of the other failures given by different scholars in the literature.

2.8.2. Causes of project failure

In the 1990s and the 2000s, academics and practitioners showed interest in determining the causes of software development project failures. Such interest started as early as the 1970s. The same issue was again a subject of discussion in the 1980s, picking up momentum in the 1990s and 2000s.

As mentioned earlier, literature on this issue dates back to the 1970s (Bostrom & Heinen, 1977). Lyytinen and Hirschheim, (1987) and others picked it up in the 1980s, Ewusi-Mensah and Przasnyski (1991) and others further investigated the issue in the 1990s and many other researchers in the 2000s. In the 1990s, Ewusi-Mensah and Przasnyski (1991) found that organisational politics and disagreements were to blame for project abandonments. They also indicated that other issues such as end-user resistance and IS professionals can contribute to project abandonments. Jones (1995) state that most software development failures are caused by poor project management and poor management practices. Jones (1995, p. 87) states that project management problems include failure to use automated estimating tools and

automated planning tools; failure to monitor progress or milestones; failure to use design reviews, and failure to use code inspections. Poor management practices can cause the following failures: inadequate design and specifications and the failure to use formal configuration control. In his study of project escalation (projects that are continued in spite of the fact that they don't perform well), Keil (1995) found four causes of project failure. They are project related factors, psychological, social and organisational factors. According to Keil (1995), project factors are objective features of the project as perceived by management (p. 422). Psychological factors have to do with management convincing themselves that problems encountered by the project are only temporary and that continued investment in the project will eventually yield positive results (p. 423). Social factors include rivalry among groups and departments of the organisation, both internal and external (p. 423). Organisational factors involve the structure and politics surrounding the project.

Linberg (1999) conducted a study of what software developers perceived as failure and success in projects and the causes of success and failure. The study found a number of factors that contribute to software project failure. They are poor management and poor marketing research. The former entails management setting unrealistic deadlines and putting pressure on the development team to meet those deadlines. The latter entails poor awareness of customer requirements for the system.

The issue of failed projects persisted in the 2000s. In their study of critical success and failure factors in IS, Poon and Wagner (2001) identified two critical failure factors in IS: management and organisational issues. They reported that management could lead to an IS system failing if all strategic decisions concerning the system are done at management level. They state that organisational politics occur when members of the organisation do not want to use the system because they fear losing their influence on top management (p. 406). Yeo (2002) reports on similar causes of project failure. Yeo (2002, p. 243) reports that software

project failures are caused by managerial and organisational factors and project management factors. Yeo (2002) states that managerial and organisational factors include issues such as "hostile company culture, improper reporting structure, political pressures, vested interests, influences, and inappropriate level of management commitment" (p.243).

In his 2003 book Ewusi-Mensah presents several examples of software project failures and concludes that they fail because of managerial and organisational problems. Bupa (2005) reported several causes of systems failure in the United Kingdom's public sector. Among those causes are poor understanding of the IT industry in the public sector, shortages of staff in the project and programme management skills to deliver big programmes, suppliers over-promising what cannot be delivered to win contracts, failure to engage with those who will use the new systems, and other stakeholders' issues. (p. 15). Charette (2005, p. 45) lists a number of factors that cause IS failure. They are unrealistic or unarticulated project goals, badly defined system requirements, use of immature technology, inaccurate estimates of needed resources, sloppy development practices, stakeholder politics, commercial pressures, Charette (2005) groups these causes into three broad categories; technical, project management and business decisions. Charette states that chief among the business factors are competition and the desire to cut costs. IS projects are not viewed by management as investments but as pure costs that must be controlled. Project management factors have to do with project managers under-estimating costs, and timelines. Technical problems arise from the fact that organisations are sometimes obsessed with using the latest technology even if the technology is not suitable for the project.

Kappelman, McKeeman and Zhang (2006) list as many as 53 early warning signs of a project deemed to fail. They group these early warning signs into three broad risk categories: people, process and product risks. They state that people risks are risks associated with a project lacking top management support, having a weak project manager, a project with no

stakeholder input/involvement, team members lacking skills, and subject matter experts that are overscheduled (p. 34). Process related issues involve the lack of resources, communication, proper requirements, change control, and tight schedules (p. 34). McManus and Wood-Harper (2007) found similar causes of software project failure as previous authors. McManus and Wood-Harper (2007, p. 42) divide reasons for software project failure into three broad categories: business, management and technical. Business reasons include among others poor requirements management and higher costs of capital. Management issues include a lack of management judgement and insufficient risk management. Technical issues include inappropriate architecture and inappropriate technology.

El Emam and Koru (2008) conducted a world survey on software development failures. They covered countries such as Australia, Canada, India, the United States of America (USA) and others. They found a failure range of 11.5-15.5%. They found that major failures are caused by management problems (senior management not entirely involved), too many requirements and scope changes, spending over budget and a lack of technical skills (p.88). Tarawneh, Al-Tarawneh and Elsheikh (2008) conducted a study of software development project success and failure in Jordan. They found that the main cause for project failure is misunderstood user requirements, which are sometimes changed during the course of the project.

Attarzadeh and Ow (2008) also surveyed students' projects to determine causes of their failure and success and compared the results with the Standish Group reports. Attarzadeh and Ow (2008, n.p.) found four broad factors that lead to software project failure. They are human, implementation, and planning and estimation factors. Human factors include poor communication and poor project management skills. Implementation factors include project and requirement changes, poor testing and incorrect development methodology being used. Planning and estimation factors include poor planning for the whole project and poor cost planning.

Tsirakidis, Kobler and Krcmar (2009) found that software development projects fail because of inconsistency in methodological development approach and information hiding in separate mailing lists. Their study was based on an agile open source team at a case organisation. Cerpa and Verner (2009) investigated 70 failed software development projects to determine what caused failure in those projects. They found two major factors that cause failure: people and project management factors. People factors include lack of rewards for overtime for staff, an unpleasant experience working on projects, an aggressive schedule that affects staff motivation, and a negative schedule that affects the team members' lives. Project management factors include not conducting post-mortem reviews after each phase or after the completion of the project, and poor risk assessment for the whole project. Lehtinen *et al*. (2014, p. 624) categorised the causes of project failure into four broad categories: people, methods, environmental and tasks. People cause failure because of a lack of skills and motivation. The development methods that organisations employ in developing software can also cause failure. For example, excluding users at the development stage can cause failure. An un-conducive environment also causes failure.

Failure to learn from past mistakes has also been reported in the literature as the main cause of software development failure. This problem has led to software organisations adopting organisational learning principles in the 2000s. Conferences and workshops were held all over the world to encourage software organisations to learn from their mistakes in order not to repeat them in future. Ewusi-Mensah and Przasnyski (1991) concluded that IS projects fail (are abandoned) because organisations fail to learn from their past mistakes. They state that "organisations are cursed to repeat past mistakes because no effort is made to understand what went wrong nor are attempts made to learn from past mistakes (p. 10)". Lyytinen and Robey (1999) share the same sentiments that these failures are caused by the software organisations' failure to learn from their past mistakes. They state that organisations have

failed to learn from their past experiences in such a way that they have learned to fail (p. 86). They give a couple of examples of organisations that have failed projects because they never learned from their past experiences, thus repeating the same mistakes. Cerpa and Verner (2009, p. 130) also noted that software has been developed since the 1960s but we have not learned enough to ensure that software development projects are successful. They are of the opinion that few post-mortem interviews are conducted after projects are completed, which denies organisations the opportunity to learn from past projects. Their words are echoed by Linberg (1999, p. 190) who says that most often a failed project causes a learning failure because no experience is transferred to the next project. Bupa (2005, p. 15) also states that, despite the publicity of failed IS projects, there seem to be no lessons learned from such failures, "but repeated ad nauseum in perpetuity." Dingsøyr, Moe and Nytrø (2001, n.p.) state that small and medium-sized software development companies experience the same problems repeatedly, but fail to learn from their own mistakes as well as their own successes.

The failure of organisation to learn from past mistakes seems to be a recurring problem. Almost every year there are stories about big and small projects failing. Cecez-Kecmanovic, Kautz and Abrahall (2014, p. 561) state that there is a substantial body of literature available on this subject, but it seems this knowledge has not made any difference. We still have a high failure rate. Charette (2005, p. 48) notes that "the reasons that software projects fail is well known and have been documented in countless articles, reports, and books and yet, failures, near-failures, and plain old bad software continue to plague us, while practices known to avert mistakes are shunned. Even organisations that get burned by bad software experiences seem unable or unwilling to learn from their mistakes." The same sentiments are shared by Nelson (2005, p. 62) who states that "failure to learn from mistakes has consistently been a major obstacle to improving IT project management". Boddie (1987) states that failure needs

to be an opportunity to learn something rather than embarrassing moments to be quickly forgotten.

The literature identifies many different causes of software project failure. These causes can be grouped into four broad categories: management, technical, organisational and external. Failure to learn from past mistakes and successes featured strongly in the literature as the main cause of software project failures. This issue has prolonged the problem because past mistakes are repeated over and over again. On the other hand, if organisations could learn from past failures, this problem could have been eliminated. The study will adopt the three broad definitions of failure: management, technical and organisational. Management failures include a lack of support from top management, and also project management issues. Technical causes are associated with a lack of technical skills and incompatible systems. Organisational skills include politics and poor staffing. External factors have to do with clients rejecting the system. Table 5 shows the conceptual and theoretical framework.

Table 5: Conceptual and theoretical framework

Knowledge management concepts		
Concept	**Definition**	**Research questions to be addressed**
Acquisition	The activity of accepting knowledge from the external environment and turning it into a commodity that can be used in the organisation (Holsapple & Joshi, 2002; Bukowitz & Williams, 1999).	RQ 2 -What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Application	The activity of applying available knowledge in organisational routines and processes (Holsapple & Joshi; 2002; Gottschalk, 2007).	RQ 2 -What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Creation	Involves developing new knowledge and replacing existing knowledge with the new one (Nonaka & Takeuchi, 1995; Pentland, 1995; Mitchell & Boyle, 2010)	RQ 2 - RQ 2 -What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Organisation	Cataloguing, indexing, filtering, and the codification of knowledge (Awad & Ghaziri, 2004; Samoilenko & Nahar, 2013).	RQ 2 -What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Sharing	The process in which knowledge is communicated to other people (Li & Gao (2003; Fernandzez, Gonzalez, & Sabherwal; 2004).	RQ 2 -What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Storage	Gathering the relevant knowledge and keeping it for the use in complex systems development tasks and achieving organisational goals (Andrade *et al*., 2013; Samoilenko & Nahar, 2013).	RQ 2 -What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
Software development failure concepts		
Failure	A system that does not meet the client's expectations (Lyytinen & Hirschheim, 1987).	RQ1 - What is the form and source of software development failures experienced by software development SMMEs?
Types of failure	Challenged or failed outright/abandoned (Heeks, 2002; The Standish Group's CHAOS Manifesto, 2013).	RQ 1- What is the form and source of software development failures experienced by software development SMMEs?
Causes of failure	Management, technical, organisational and external factors (Ewusi-Mensah and Przasnyski, 1991; Yeo, 2002; McManus and Wood-Harper, 2007)	RQ 1 - What is the form and source of software development failures experienced by software development SMMEs?

2.9. Learning organisations

The study views software development organisations as knowledge intensive. A knowledge intensive organisation is a learning organisation hence the importance of defining the concept of learning organisations and learning software organisations.

Easterby-Smith and Lyles (2011) state that the notion of 'the learning organisation' was developed in the 1980s by scholars such as Garratt, Peler, Boydell and Burgoyne and de Geus, but it was not frequently used until it was popularised by Peter Senge in his book called *The Fifth Discipline: the art of practice of the learning organisation*. Peler, Boydell and Burgoyne (1989) used the term 'learning company'. It is an organisation which facilitates the learning of all its employees or members and continuously develops itself (p. 2). They further state that a learning company has four characteristics: i) it has a climate in which individuals are encouraged to learn and to develop their full potential, ii) it extends its learning culture to include all stakeholders, iii) it promotes human capital development policies to enable organisational learning, and iv) has a continuous process of transformation to take advantage of individual learning to change the operations and aspirations of the organisation (p.3-4). This kind of organisation promotes individual best practices and uses them within the organisation. Armstrong and Foley (2003, p.74) define it as an organisation as an organisation that promotes the learning of its individual members to create valued results.

There has been confusion between organisational learning and the learning organisation concepts. These concepts are sometimes used interchangeably. According to DiBella (1995, p.287) the difference between the two concepts is that organisational learning is about the learning processes that can take place at different levels within the organisation, whereas the learning organisation is a concept used as a metaphor to describe a type of organisation that

learns. Ortenblad (2001, p. 126) says that the learning organisation is a form of organisation yet organisational learning is an activity or process of learning in the organisation. The study adopted the organisational learning concepts because it is concerned with how organisations learn, that is, the learning processes at individual, group and organisational levels. The study is not concerned with the characteristics of an organisation that learns. In the software industry, learning organisations are called learning software organisations. A brief discussion of learning software organisations is given below.

2.10. Learning software organisations

Software organisations are knowledge intensive organisations that build and deploy software. These are not perfect organisations because they also suffer from learning disabilities that lead to time overruns and increased costs. A new type of software organisation called 'learning software organisation' has been conceptualised to address the learning problems software organisations face. Birk, Dingsøyr, Lindstaedt, and Schneider (2006, p.61) define a learning software organisation as "an organisation that develops or maintains software and intentionally acts as a learning organisation". They further state that this kind of organisation relies heavily on individual learning and individual competencies. The individual learning and competencies are then institutionalised. Learning software organisations have been created to foster learning at organisational level and to capitalise on the knowledge assets of the software organisation (Althoff, Bomarius, & Tautz, 2000). Feldmann and Althoff (2001, p.2) state that a learning software organisation promotes the management of its knowledge, turns it into intellectual property and turns it into market shares and profits. They further state that this kind of organisation continuously fosters learning and sharing of experiences within it. A learning software organisation aims to improve communication (group learning), which enables organisation-wide learning. This is done by documenting knowledge and storing it for future use. In summary, learning software organisations apply the principles of the

learning organisation in the software development context. They strive to learn continuously from their experiences to stay competitive. They learn by facilitating individual, group, and organisational learning. Chouseinoglou and Bilgen (2012, p.252) define a learning software organisation as an organisation

that learns within the domain of software development, evolution and application where the objects of learning can consist of models, knowledge and lessons learned related to the different processes, products, tools, techniques and methods applied during the different stages of the software development process.

The literature indicated that software development organisations adopted the learning concept to improve their processes. This is because the software development task is knowledge intensive and requires learning at individual and organisational levels to meet organisational goals.

2.11. Summary

This chapter discussed the concepts of the study. From the discussions, it emerged that there is a lot of debate about knowledge as a concept. There are different arguments about the nature of knowledge. Scholars argue that knowledge is created from data and information, while other claim data and information are created from knowledge. These debates have also made the definition of knowledge management difficult. As it is an accepted field of study, other scholars dismiss that notion and argue that it is a fad. This study adopted the data, information, knowledge perspective and accepted Nonaka and Takeuchi's (1995) classification of knowledge into tacit and explicit. In short, the study accepts that there is tacit and explicit knowledge and that knowledge is derived from data and information. The chapter indicated that there are three knowledge management perspectives: the codification and personalisation, the schools of thought and the lifecycle perspective and this study

adopted the lifecycle perspective. A conceptual and theoretical framework was developed from 15 lifecycle frameworks found in the literature. The concepts of the framework are knowledge acquisition, creation, storage, organisation, sharing and application. The chapter also conceptualised software development failures, their types and causes. The chapter revealed that failures are abandoned or challenged projects and they are caused by human, organisational and technical factors. After the concepts were defined and adopted, a unified conceptual and theoretical framework to inform the study was developed.

CHAPTER THREE
LITERATURE REVIEW

3.1. Introduction

This chapter reviews the literature on knowledge management in software organisations. The literature review is based on the research questions posed in the study. The chapter discusses software development failure rates and the three main intervention strategies. To address software development failures, software organisations have been trying to adopt SPI, knowledge management and agile methods. SPI is a set of practices that software organisations are encouraged to adopt in order to develop high quality software. Agile methods are new software development methods that are believed to be more flexible than traditional methods. These new methods are assumed to enable software organisations to address the problems they face. Because software development is knowledge intensive, software organisations have also tried to adopt knowledge management practices. These are practices that enable the organisations to acquire, store and use knowledge when they develop software. The literature review discusses this in detail.

The chapter is structured as follows. The first to be discussed is software development failure rates. That discussion is followed by failure intervention strategies. The first intervention strategy discussed is SPI and its models, the capability maturity model integration (CMM/CMMI). The discussions will be based on how SPI is used to address software development project failure issues. The last intervention strategy to be discussed is knowledge management. Knowledge management discusses issues such as how software organisations acquire, create, store, organise, share, apply and learn from knowledge resources at their disposal and how these processes address software development failures.

3.2. Failure rates

The Standish Group is well known for having documented software development project failures since 1994. Over these years, the reports have been portraying a negative image of software development projects. According to the reports, since they started documenting project failure, there has never been a year in which more than 50% of the projects were successful. A summary of the reports from the years 2011-2015 is shown in Table 6.

Table 6: *The Standish Group's CHAOS reports 2011-2015.*

Projects' outcome	**Year**				
	2011	2012	2013	2014	2015
Successful	29%	27%	31%	28%	29%
Challenged	49%	56%	50%	55%	52%
Failed	22%	17%	19%	17%	19%

(Source: table adapted from Hastie and Wojewoda, 2015)

The results of the Standish Groups' reports over the past decade show that less than 40% of the all software development projects are successful. This means that about 60 % of all projects are either abandoned or challenged. Although the Standish Group's reports have been cited extensively in the software engineering literature, there have been criticisms of the results. For example, the high failure rate is questioned by Jorgensen and Molokken-Ostvold (2006). They assert that this might be caused by methodological and analysis flaws. Glass (2002) also questioned the validity of the Standish Group's results based on the fact that they are biased towards failed projects and that the Standish Group has failed to reveal their data sources. But these reports continue to be influential in the IS field.

In South Africa the technology company ITWeb has been conducting similar studies to those of the Standish Group since 2009. The results of the South African studies are shown in Table 7.

Table 7: ITWeb project failure reports 2011-2013.

	2011	2012	2013
Successful	12%	11%	11%
Failed	10%	6%	6%
Within budget	18%	15%	15%
Ready on time	13%	10%	16%
Full functionality	24%	23%	32%
Overall quality	17%	21%	20%

(Source: ITWeb software development reports 2010-2013)

The results show that over three years, less than 13% of projects were absolutely successful, less than 20% were within budget, less than 17% were on time, less than 33% had full functionality and 20% or less emerged in good quality. These results are almost the same as the Standish Group's. They also paint a damning picture of a software industry in crisis, but this is in contrast to what Smith (2002, p. 2) found previously in South Africa, namely that 53% of software project were successful, with time and cost overruns amounting to only 27% and 33% respectively and functionality as high as 84%.

Therefore, the question asked is what causes such high failure rates in software development? The next section will discuss the possible answers to the question.

3.3. Intervention strategies

The literature indicates that there are three main intervention strategies used to address the software development failure problems; SPI and its sub-frameworks, agile methods and knowledge management. This section discusses the intervention strategies. First to be discussed is SPI, followed by agile methods and lastly knowledge management.

3.3.1. Software Process Improvement (SPI)

As stated in earlier sections, software development problems have been present ever since software was first developed. In 1982, the USA department of defence recognised the urgency to deal with this problem and formed a task force to address it. That led to the development of the SPI framework. SPI aims to address three concerns: "the management of improvement activities, the approach implemented to guide the improvement initiatives, and the perspective adopted to focus attention on the improvement goals" (Bellini & Storto, 2006, p. 1257). Lehtinen *et al*. (2014, p. 623) state that SPI was created to address project failures in software development by improving product quality, lowering development costs, and shortening the time to market. SPI as a framework led to the development of other models such as CMM (Humphrey, 1992).

The CMM strives to help software organisations produce high quality products. It addresses issues such as software process, quality management, and process improvement (Humphrey, 1992, p.1). Bellini and Storto (2006, p. 1258) define it as a model for judging the maturity of the software development processes in organisations and to identify key processes that are important in the development process. This model has five levels that software organisations can take in the development of software products. Each level represents how well a software company develops software. Other SPI models developed over the years include CMMI, ISO

9001:2000, ISO/IEC 12207:2004, ISO/IEC 15504:2004, and SPICE (ISO/IEC, 15504:1998) (Pino, Garcia & Piattini, 2008, p. 238).

The software process improvement models do not seem to have addressed the issue of software development failure, especially in small software organisations. Saranya and Kannan (2013, p.1) state that owing to the challenges that SPI models still pose to small software enterprises, doubts have been raised on the application of such models to software organisations. They list six challenges that prevent SPI models from being implemented in software organisations: high individual dependence, overloaded persons, human factors, small number of projects, importance of customer communication, and funding constraints. Habra *et al*. (2008, p. 42) concur and state that these models were not written for organisations with fewer than 25 people, thus making it difficult to apply in small organisations. They further state that such models are not suitable for small companies because of their organisational structure and the costs that go with the implementation of such frameworks.

Niazi (2009, p. 25) states that SPI implementation is usually hindered by time pressure, lack of support, lack of SPI awareness, inexperience, lack of resources, lack of staff, lack of defined SPI implementation, methodology and organisational politics. Chouseinoglou and Bilgen (2012, p. 252) acknowledge the role of SPI models in managing software engineering knowledge, but indicate that such models have not been successful because they do not clearly indicate the knowledge that needs to be managed and explain how, when, where it should be managed, or by whom and for whom.

3.3.2. Agile methods

Another intervention strategy developed to address the problems encountered in software development was the adoption of new software development methods called agile methods. In the year 2001, a group of 17 practitioners met and agreed on a new software development method called 'Agile methods'. According to the Agile Manifesto (2001), agile methods value individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation and responding to change over following a plan. Boehm (2002) states that agile methods entail practices such as short iterations, frequent releases, peer review, on-site customer participation, and simple design.

The main reason agile methods were developed was to counter the challenges faced by traditional software methodologies (Cao & Ramesh, 2007, p. 41) which were causing project failures. Ferreira and Cohen (2008, p. 48) state that the high software development failure rate is attributed to traditional software methodologies (the waterfall method). They further state that agile methods have emerged to address such limitations. The same sentiments are shared by Serena (2007, p.4) who states that traditional methodologies have failed to completely eliminate software development challenges and thus organisations have adopted agile methods. These words are echoed by Maher, Kourik and Chookittikul (2010, p. 300), who state that the flexibility of agile methods aims to address the problems faced by the software industry such as changing delayed deadlines, cancellations, and applications failing to match customer requirement.

The literature indicates that the adoption rate of agile methods in the world is increasing steadily. Maher *et al*. (2010) reported that a number of organisations have adopted agile software development methods in Vietnam. Rodríguez, Markkula, Oivo and Turula (2012) also conducted a similar study in Finland. Their aim was to investigate the adoption of agile

methods in Finland. They concluded that more than 58% of surveyed software organisations have adopted agile methodologies (p. 139). Salo and Abrahamsson (2008) investigated the adoption of agile methods (Extreme Programming and scrum) in several European countries. They discovered that these agile methods are widely used by European software development companies. For example, they reported that 54% of European organisations use extreme programming (p. 63).

In the South African context, very few studies are available. Ferreira and Cohen (2008) investigated the effects of five characteristics of agile systems development. Their results showed a strong positive effect of agile practice on stakeholder satisfaction with both development process and the project outcome. Noruwana and Tanner (2012) investigated the phases South African organisations go through whilst adopting agile methods and the disparities between agile prescriptions and the way they implement agile methods. They discovered no structured processes that these organisations go through before implementing agile methods; also that organisations go through various phases when trying to adopt agile methods. Ramnath (2010) discovered that South African organisations employ mostly agile methods in their software development because of the small sizes of the development teams and the complexity of the changing requirements and the short duration of projects.

There are several benefits that have been brought by agile methods and reported in the literature. Begel and Nagappan (2007) found that agile methods improved communication in teams, allow quick release of software products, and increased flexibility in design. Vijayasarathy and Turk (2008, p. 5) found the following benefits of agile methods in software development: the ability to be flexible and to deliver quality software that meets customer needs faster, improved software quality, increased flexibility in development, faster delivery, lower development costs, and more reusable code. Ambler (2008) found similar findings. Ambler found that the adoption of agile methods has improved software quality, reduced

costs, increased business stakeholder satisfaction, and increased overall software productivity. VersionOne (n.d.) indicated that agile methods have significantly reduced the risk associated with software development, maximised value through the development process, enabled teams to easily adapt to changing environments, and improved visibility in the software development process.

Agile methods are also presumed to enhance knowledge management in teams and organisations. Šmite et al. (2017) state that agile knowledge networks benefit individuals and teams when solving complex, unfamiliar, or interdependent tasks. Chau, Maurer and Melnik (2003) analyse a number of knowledge management practices adopted by agile teams. Among them are project documentation (although agile methods prefer less documentation), the use of informal training methods, emphasising on knowledge sharing between stakeholders and users during the design phase of projects, and continuous learning. Dorairaj, Noble and Malik (2012) state that agile team members practice daily scrums, sprint planning and sprint reviews and project retrospective meetings to share knowledge. Neto (2016, p. 44) list a number of knowledge management practices adopted by agile teams. They include daily scrum meetings, iteration planning and reviews, pair programming, retrospectives and refactoring.

Razzak and Ahmed (2014, p. 1434) state that agile teams use communities of practice, pair programming, scrum boards, and customer collaboration for knowledge creation and sharing purposes. Sivanantham (2012, p. 3) share similar sentiments and state that agile teams use release and iteration planning, pair rotation and pair rotation, daily scrum meetings, cross-functional teams and retrospectives to create and share knowledge. Other knowledge management practices adopted by agile teams as indicated by Razzak and Ahmed are technical forums, workshops and seminars, and knowledge repositories. Sidenvall (2017, p. iii) state that agile teams create and share knowledge "through having members work closely

to each other and share experiences, and through practising their skills in daily work, with help from each other when necessary". Apart from that, they practice knowledge management by giving teams time off to innovate, through presentations and reading tips, and communities of practice (Sidenvall, 2017, p. iii). Neto (2016, p. 49) state that extreme programming (XP) is the most used agile method in knowledge management context, followed by scrum.

Agile methods are concerned mainly with tacit knowledge as compared to traditional methods who depend very much explicit knowledge (documentation). Razzak and Ahmed (2014, p. 1434) make an example of innovation boards as a type of tacit knowledge that is used in agile teams. Similar sentiments are shared by Sivanantham (2012, p. 4) who state that agile teams prefer direct communication through iteration planning, retrospectives and daily scrum meetings. All these practices rely on knowledge sharing by social interaction (that is, sharing through tacit knowledge). Ryan and O'Connor (2013, p. 1614) state that agile teams acquire and share tacit knowledge "directly through good quality social interactions and through the development of a transactive memory system with quality of social interaction playing a greater role than transactive memory". They continue to state that "transactive memory system and team tacit knowledge predict effectiveness but not efficiency in software teams" (p.1614).

The studies above indicate that agile methods, especially XP and scrum use a number of knowledge management practices to enhance their development tasks. The most common ones are communities of practice, information communication and daily scrum meetings. The studies also reveal that agile methods are more concerned with sharing tacit knowledge than explicit knowledge.

However, agile methods have problems of their own. One major problem is that they cannot be scaled to larger projects with larger teams (Begel & Nagappan, 2007; Vijayasarathy & Turk, 2008; Maher *et al*, 2010; Rodriguez *et al*, 2012). Begel and Nagappan (2007) also found that agile teams have difficulty blending with non-agile teams. Vijayasarathy and Turk (2008) also found that agile methods are not suitable for distributed teams and that they are difficult to learn.

As much as the literature somehow provides evidence that agile methods have improved software development by reducing costs and producing quality software, there is no evidence that these methods have completely addressed the issue of project failures. Current statistics still reveal that software project failure rates are still very high.

Knowledge management is another intervention strategy that software organisations are trying to adopt to address software project failure problems (Faher & Gabor, 2006, p. 252). Knowledge management is also believed to be capable of addressing the learning problems that software development organisations face. The following section discusses knowledge management in software organisations; that is, how it has been applied and whether it has helped organisations.

3.3.3. Knowledge management

The early 2000s saw software organisations adopting knowledge management practices to manage information and knowledge using different techniques and technologies such as codifying knowledge and building knowledge databases, promoting networks of experts within the organisation, and enhancing software development processes (Bjornson & Dingsoyr, 2008). This section reviews literature on how knowledge management practices have been applied in software development and in software organisations. The structure of this section is as follows: the first subsection discusses how knowledge management is

applied in software organisations in terms of the knowledge management life-cycle developed in Chapter Two. That is, how knowledge is created, acquired, shared, stored, organised, and applied in software organisations and whether or not knowledge management has benefited software organisations. The focus is on organisational learning. That subsection is followed by discussions on how knowledge management is applied generally in different contexts in software engineering, namely knowledge management practices in the systems development lifecycle (SDLC), knowledge management in software teams, and knowledge management practices by software organisations.

The following section reviews literature on knowledge management practices according to the conceptual and theoretical framework in Chapter 2.

Adopting Earl's (2001) framework, Bjornson and Dingsoyr's (2008) literature review categorises knowledge management studies in software development into three broad schools: behavioural, economic, and technocratic schools. This subsection takes a different angle. It focuses on the seven processes as defined in the conceptual and theoretical framework discussed in Chapter 2: knowledge creation, acquisition, storage, organising, sharing, application, and learning.

3.3.3.1. Knowledge acquisition

According to Holsapple and Joshi (2002), knowledge acquisition is the activity of accepting knowledge from the external environment and turning it into a commodity that can be used in the organisation. Several studies are found in the literature describing how software organisations acquire knowledge and the type of knowledge they acquire. Gendreau and Robillard (2013) discuss knowledge acquisition activities in a software development environment. They state that knowledge is acquired from internal and external. Examples of external sources are the Web, technical documentation, a book or paper. Internal knowledge

is the tacit knowledge in the developers' heads (p. 2). Their conclusion is that the acquisition of information, which they call learning, leads to improved knowledge (p. 9). Ryan and O'Connor (2013) took a different direction and conducted a study that sought to investigate how development teams acquire and share tacit knowledge. Their conclusion was that, in software development teams, tacit knowledge is acquired and shared by social interactions and by developing transactive memory system (p.1622). This system is based on the notion that long-tenured groups members tend to rely on one another to acquire, process, and disseminate information from unique knowledge domains (p.1615). Trimble's (2000) study is also about knowledge acquisition processes, but the study focuses on how these processes contribute to reducing the risk involved in software development. The five processes are collecting software metrics, communication, creating approaches to meetings, conducting observation sessions, and knowledge elicitation from experts (Trimble, 2000, p.175-178).

The conclusion of the study is that structured knowledge acquisition approaches can facilitate reliable knowledge acquisition and development. Savolainen and Ahonen (2015) noted the difficulties project managers encounter when acquiring knowledge between the implementation and the sales phases of a software development project. They state that this is usually caused by changing project managers; that is, using a different project manager in sales and a different one in the implementation phase. They claim that this causes a knowledge acquisition gap.

The literature on knowledge acquisition in software development environment indicates that most of the studies conducted focus on knowledge acquisition processes; how knowledge is acquired from different sources.

3.3.3.2. Knowledge creation

Knowledge creation involves developing new knowledge and replacing existing knowledge with the new knowledge (Pentland, 1995, p. 3). Knowledge creation is an area that has been investigated extensively in software development. The interest emanates from the assumption that software development is a knowledge creation process (Bailin, 1997). Studies in this area focus on many aspects of knowledge creation. Some focus on knowledge creation processes in software development, while others focus on the end product as knowledge creation, and many others focus on knowledge creation activities of software developers and knowledge creation in software organisations. Arent and Norbjerg (2000) studied SPI projects from the knowledge creation perspective. Their study revealed a number of knowledge creation activities such as the creation of tacit and explicit knowledge. They state that tacit knowledge is created through the sharing of ideas between development and SPI teams. Explicit knowledge is created by the documentation of ideas. They discovered that knowledge is created in three groups: in the development group, in the SPI group and in the interaction of both groups (p. 9-10). Their conclusion was that SPI processes are knowledge creation activities.

Kess and Haapasalo (2002) investigated knowledge creation through a software project review process, where they came up with a tool to improve the software development process. The tool consists of five steps (p. 54): 1) The definition of review practices, 2) Training of all software development professionals, 3) The definition of all intermediate products in a project, 4) The definition of all metrics, and 5) The definition of the process to utilise the information gathered from the reviews. Their conclusion was that these are knowledge creation steps necessary to improve software development. Klint and Verhoef (2002) studied how knowledge management principles could be applied in knowledge creation in software development organisations. They concluded that software "maintenance

and renovation can easily be seen as knowledge creation processes where tacit knowledge is made explicit" (p.157). Linden and Cybulski (2009) investigated pattern mining as a knowledge generation process. Pattern mining is a software development methodology useful in acquiring, validating, recording, enriching, and sharing problem solving knowledge. Pattern mining facilitates the capturing and dissemination of best practices and encourages continuous improvement of captured knowledge (Linden & Cybulski, 2009, p.1). Their conclusion was that pattern mining is a knowledge creation activity. Their study further developed Wickramasinghe and Lichtenstein knowledge creation theory. Using Luhmann's theory of social systems, Morner and von Krogh (2009) explored knowledge creation processes and the conditions under which successful knowledge creation takes place in open source software development teams. They found that knowledge is created when tacit and explicit knowledge interact. Tacit knowledge could be the developers' knowledge about software development, and explicit knowledge could be the code stored in emails and other databases. They proposed that knowledge creation could be stabilised under three conditions perceptibility, systemic memory, and modularity.

Wan *et al.* (2010) investigated knowledge creation in requirement elicitation process (REP). Their study focused on how knowledge is created in REP. They concluded that a friendly project environment, top management support, and clear project plans play a huge role in knowledge creation (Wan *et al*., 2010, p. 494). Their study led to the development of a knowledge creation model in REP which is based on the SECI model developed by Nonaka and Takeuchi in 1995.

3.3.3.3. Knowledge sharing/transfer

According to Major and Cordey-Hayes (2000, p. 412) knowledge is transferred when it is conveyed from one place or person, to another. Studies in this area focus mainly on knowledge sharing activities, challenges and knowledge sharing channels between teams and organisations. A few studies focus on knowledge sharing between individuals (software developers). This is an area that needs to be explored further. Holz and Maurer (2002) in a study on distributed agile teams found that the teams shared knowledge mostly via the Internet, using Internet platforms such as newsgroups and dedicated websites. Teams also used technical reports and databases of lessons learned to share knowledge. Lakalu *et al.* (2010) developed a knowledge sharing framework for open source software development environments. Their study was inspired by an earlier survey that uncovered knowledge sharing difficulties in systems development. The framework provides a mechanism that supports knowledge sharing throughout the development process. It further gives opportunities and guidelines for communities of practice to share their knowledge (p. 84-85). Chan and Husted (2010) investigated knowledge sharing behaviour in open source software development organisations. Their aim was to investigate the effect of dual allegiance to knowledge sharing. They discovered that dual allegiance plays a role in determining individual sharing behaviour in collaborative environments such as open source communities.

Apart from studies that focus on good knowledge sharing activities in software organisations, other studies focused on the barriers to knowledge sharing. Kukko's (2013) study focused on knowledge sharing barriers in a software development organisation. Kukko (2013) found that knowledge sharing barriers are found at three levels in organisations: at individual, organisational and technological levels. Time is the most serious knowledge sharing barrier at individual level. Other barriers found at individual level are a lack of trust, lack of social networks, and low awareness of the value of knowledge possessed. At organisational level, a

disconnect between knowledge sharing purpose and organisational goals and neglect of managerial communication about the benefits of knowledge sharing were found to be knowledge sharing barriers. Lack of time is also a barrier at the technological level. Nidhra *et al*. (2013) conducted a similar study; they investigated knowledge sharing challenges and their mitigation strategies in global software development companies. They found 60 challenges and 79 mitigating strategies, and grouped them into three broad categories: technology, personnel, and project challenges (Nidhra, *et al.*, 2013, p. 333). Personnel challenges include staffing, trust, language, cultural differences, and personal attributes (p.341). Project factors are communication, infrastructure, novelty, requirement specification, temporal distance, extra costs, changing vendor, project deadlines, and communities of practice (p. 342). Technology factors are support tools and organisational memory system (p. 344).

3.3.3.4. Knowledge storage

The knowledge management literature provides a comprehensive overview of how organisations in different fields store knowledge. The literature indicates that knowledge is stored in many formats. For example, knowledge is stored in books, the Internet, in peoples' heads and organisational databases. In the software development field, knowledge storage is regarded as a very important activity. This is because knowledge stored is to be used in the future. Studies have been conducted in the software development field to explain how knowledge is stored in software organisations. These studies address knowledge storage issues such as knowledge storage technologies, knowledge storage processes and knowledge storage architecture. For example, Antunes, Seco and Gomes (2007) describe how knowledge is stored and reused in software development organisations by means of ontologies. Their study describes how a knowledge storage system called Semantic Reuse System is used to store and retrieve organisational knowledge. The system allows software developers to

capture, store, and retrieve knowledge. It has searching, indexing modules for easy access to knowledge. García *et al.'s* (2011) study looks at the development and effectiveness of knowledge storage process access library systems (a PAL-Wiki) and how it facilitates learning in agile software development. According to García *et al.* (2011, p.834) the system helps by capturing and using software development process artefacts. Artefacts such as manuals, and lessons learned, are captured and stored in the system. Developers then use the stored artefacts, thus saving time and effort for them. Developers also learn in the process.

Samoilenko and Nahar (2013) conducted a literature review on the difficulties that high tech organisations face when storing and retrieving knowledge. The review focused on IT tools that could be used to address the challenges that high-tech organisations face. They proposed a framework of tools that could be used by software organisations to store and retrieve knowledge. The framework indicates that knowledge storage and retrieval involves the following processes: assimilation, codification, formalisation, and formatting (p.1359). The study then presents different tools such as knowledge databases (data warehouses and knowledge repositories), electronic bulletin boards (message boards and computer forums), enterprise resource planning (ERP) systems, multimedia databases, content management systems, distributed storage systems, Wiki systems and blogs for knowledge storage and retrieval.

Jansma and Means (2012) report how the National Aeronautics and Space Administration Agency (NASA) store its systems engineering knowledge of all types. They reveal that NASA stores its systems engineering knowledge in a database called NASA Systems Engineering Body of Knowledge (SEBoK). They state that the system captures and stores all the professional knowledge within the agency. This knowledge is then used by different departments within the agency for systems development tasks.

Other organisations use the concept of experience factories to store knowledge. The concept 'experience factory' was coined by Basili *et al.* (1994). This concept describes a certain type of software organisation that captures and uses knowledge gained from past projects (experience) and uses it in current and future software projects. The 'experience' is stored in knowledge repositories within the firm. The main aim of the experience factory is to store, retrieve and reuse experience (Kim *et al.,* 2005, n.p.). Developers use such knowledge for their immediate and future development tasks. Studies investigating this phenomenon focus mainly on the design of experience factories. For example, Seaman, Mendonca, Basili and

Kim (1999) provide the design and architecture of an experience system. The study suggested how such a system could be designed to meet the requirements of a client. Issues such as user interfaces and retrieval systems are discussed in the study. They proposed a design and architecture that is deployable and they assumed that it could help their client. A similar study was conducted by Kim *et al.* (2005). The study proposed the design and deployment of an experience database to be used by companies who adopted SPICE. They proposed that the database could be used specifically for providing benchmarking data for SPICE assessments, tells which activities should be given priority when using SPICE and is able to discover weaknesses of SPICE.

Basili *et al*. (2007) also suggests the design and architecture of an experience factor. They suggest different tools to be used, processes to be used when using the experience factory and structure of the experience factory. The literature indicates that the area of experience factories lacks studies investigating the use and benefits of such systems. This is an area that needs attention as it is not known whether such knowledge repositories have helped organisations or not. All that we know is how to design them. Seaman *et al*. (1999) study on the design principles of an experience factory is one study that has been conducted based on the experience factory concept. They found that three principles relate to the design of such a

system: how experience can be stored and made available across boundaries, how experience can be presented to those who need it, and how experience works for the whole organisation.

3.3.3.5. Knowledge organisation

Awad and Ghaziri (2004, p.24) define knowledge organisation as the cataloguing, indexing, filtering, and the codification of knowledge. In the software engineering field, not many studies have been conducted to investigate knowledge organisation. Mat and Liu (2011) describe how a software engineering method called structured object-oriented formal language (SOFL) was used to organise software requirements analysis knowledge. According to Mat and Liu (2011, p. 65) SOFL is a tool that enables easy representation, access, search and backtracking of knowledge. The method categorises knowledge into different categories (domain, dynamic and method in this case). Detailed descriptions of each category is defined and fed into the system. Schemas are then created to further categorise the knowledge for easy access. Chow and Wong (2011) report how knowledge is organised when developing wise web software. Their study proposed a knowledge organisation framework to cater for multiple facets of knowledge. They argued that for knowledge to be organised it has to be characterised.

3.3.3.6. Knowledge application

Citing Menon and Varadarajan, Samoilenko and Nahar (2013, p.1280) define knowledge application as the actual use of knowledge for the benefit of the organisation, team and individual. They further state that it "involves various activities which are performed to facilitate effective usage of knowledge for achieving organisational goals, improving decision-making, developing complex software and systems, solving human and other problems" (p. 1280). Knowledge application is a very crucial step in the knowledge management process. It involves three activities: retrieving, processing and applying knowledge. Organisations can create and store knowledge, but if it is not used, it is useless.

Samoilenko and Nahar (2013) developed a framework of relevant IT tools, methods and strategies that have the capabilities to facilitate knowledge sharing and application in globally distributed teams of high-tech organisations. The study identified a number of tools that can be used to share and apply knowledge. Tools such as artificial intelligence, expert systems, social networks, to name but a few we identified. Baisch and Liedtke (1998) explain how knowledge extracted from a fuzzy logic system is applied in a software development environment. Their study concluded that data extracted from the fuzzy logic system is used in future projects.

3.3.3.7. Learning in software development

Learning or organisational learning has been a topic of interest in software development for a long time. This is because of the assumption that learning has positive outcomes when developing software. It is assumed that it leads to the development of quality software, within budget, and on time. The learning concept is usually associated with the learning of individuals and teams within the organisation, which then leads to the learning of the whole organisation. Concepts such as organisational learning and learning organisations and learning software organisations are widely used in the software engineering field. The literature indicates that studies in the learning field indicate that learning takes place at three levels: individual, team/group and at organisational levels. This section reviews the literature on learning at all levels in software organisations. The review starts with individual and group learning and ends with organisational learning.

Not many studies are found in the literature on individual and group learning in software engineering. A lot of literature on individual and group learning is found in the general knowledge management literature. This implies that considerable research still needs to be conducted in this area. Ras and Weber (2009) report how a learning technology (a Wiki) they call ‘the learning space’ enables individual learning in a software organisation. They state

that the learning space is generated when a user interacts with the system to access or retrieve experience. Their study revealed an improvement in learning from the use of the Wiki. This means that the Wiki promotes the reuse of experience thus improving knowledge acquisition among the developers. Knowledge acquisition improves learning.

Boden, Nett and Wulf (2010) investigated team and organisational learning in offshore teams. The teams were located in different parts of the world with different development cultures. They found that the teams learned from each other's development skills through personal meetings at different sites where developers discussed and shared development ideas. They also stated that teams learned through the division of labour which allowed them to be specialist in different tasks. This means that they learned more as they worked on specific assignments thus making them specialist in those areas. Yanghua (2008) discusses the conditions under which teams learn. The study indicates that learning is influenced by the desire to achieve goals, and that a learning motive tends to have a higher learning outcome. Lamoreux (2005) describes how agile teams learn through reflection meetings. These are meetings at which teams evaluate what happened in past projects. They are similar to post-mortem reviews. Lamoreux (2005, n.p.) states that these meetings help teams learn because individuals learn from each other by knowing each other and developing trust, which enables them to work well together. Reflection meetings also help teams learn the practices that work and those that do not, and how to adapt practices to make them work better in their environment.

Chen (2005) also conducted a study on how teams learned through the partitioning of tasks in software development projects. They identified two main learning processes, analytical and synthetic. Analytical learning is learning from knowledge gained from the analysis of written documents. Synthetic knowledge is learning by bringing together knowledge from different sources. The learning processes involved knowledge exchange between individuals performing various tasks and drawing knowledge from other teams.

The majority of the studies in this category are on organisational learning, that is, how the whole organisations learn. The studies address learning issues within the organisation and tools used to support the learning process in organisations. Organisational learning studies started getting attention in the early 1990s and they picked momentum in the 2000s. This led to the adoption of the concept learning software organisations.

In the early 2000s, Huntley (2003) studied organisational learning in open source software development. The study's aims were to investigate adaptive learning mechanisms realised by the debugging process and investigate the learning curve effects of project-specific experience on bug cycle times. The study revealed that because of learning, the complexity of the project open source was reduced. It was also revealed that adaptive learning takes place in such development exercises, because of the feedback given by both developers and users. Bellini and lo Storto (2006) investigated the relationship between CMM, knowledge management and organisational learning. Their study was based on the premise that CMM and knowledge management can influence organisational learning. Their study found five learning typologies influenced by CMM and knowledge management. They are learning by selecting, learning by doing, learning by informing, leaning by training, and learning by sharing (p, 1266). They concluded by stating that CMM can support knowledge management which in-turn supports organisational learning. Alagarasamy, Justus and Iyakutti (2006) theoretical paper addresses how knowledge management influences organisational learning in software companies. They concluded that knowledge gained through practising organisational learning helps senior managers improve the software development process.

Cao and Ramesh (2007) state the importance of learning in agile software development. They use the concepts of single and double loop learning to indicate how agile methods facilitate learning. Their conclusion is that agile software development practices promote double-loop learning because it provides a learning environment where participants can

examine and experiment with their mental models, that are implicit in what they do (p. 46). Au *et al*. (2009) investigated organisational learning in virtual open source software organisations. Their study focused on the learning effects of open-source software projects (OSS) and the factors that affect the learning process. Their study revealed that learning does take place in many open-source software projects. They further state that learning is influenced by team size, individual developers and knowledge sharing. That is, the smaller the group the quicker the learning. The participation of developers in large projects also positively influences learning.

Menolli, Malucelli, and Reinehr (2011) proposed a "semantic social collaborative environment to facilitate and enhance organisational learning within software organisations". Their paper focused on the identification of the major characteristics and needs of a collaborative organisational environment designed to implement organisational learning in software development organisations. The characteristics identified by the paper are: knowledge sharing and knowledge use by software developers, individual and organisational skills management, knowledge contextualisation and knowledge organisation according to user (p. 68). Peng, Mu and Benedetto (2013) report the role of learning in choosing open-source software licensing. They identified two modes of learning that influenced licence choice; experiential and vicarious learning. Experiential learning is learning from the experiences of individuals, while vicarious learning is learning from the experiences of others (Peng, Mu & Di Benedetto, 2013).

Experiential learning happens when first-hand experience is accumulated; social actors learn from them and apply it in future. Vicarious learning happens through co-membership. Working together with members who have acquired knowledge from past projects enables others to learn from them. Chouseinoglou *et al.'s* (2013) literature review discusses core process areas of organisational learning in software organisations. The core process areas are: obtaining, using and passing knowledge. These core process areas have core processes within them. According to Chouseinoglou *et al.*

(2013, p. 1906-1908), obtaining knowledge involves the processes of knowledge identification (discovery or capturing), knowledge acquisition (buying), and knowledge development (creation or construction). Knowledge use has the following processes: knowledge dissemination (sharing or distribution), knowledge organisation, publication, usage (application or utilisation), and knowledge integration (routines). Core processes under passing knowledge are knowledge evaluation (valuation), knowledge preservation (retention or archiving and deleting), knowledge evolution, and knowledge selling. These three major processes and core processes make up the cycle of organisational learning in software organisations.

Other studies focus on the tools that support learning in software development environments. Examples are: Garcia *et al.* (2011) which applied knowledge management principles to report about a Wiki that facilitated learning and knowledge use of software processes. Their aim was to investigate the role that Wikis played in facilitating learning in a software organisation. The results of the study revealed increased learning by the use of the Wiki by software engineers. According to Garcia *et al*. (2011, p. 847), "the development teams searched, consulted and reused the process assets found, to effectively create the work products during the training and project stages. Menolli *et al.* (2015) investigated how different technologies support knowledge sharing and learning in Brazilian software organisations. Menolli *et al*. (2015, p. 295) list the following tools as used by software organisations for knowledge management and learning purposes; repositories of shared documents, Wikis, property tools, collaborative tools, and blogs.

Another learning process widely used in software organisations is conducting post-mortem reviews, sometimes referred to as post-mortem analysis (PMA) of software development projects. Post-mortems are conducted to identify development challenges and best practices. By so doing, organisations learn. According to Dingsøyr (2005, p.293), conducting post-

mortems is a method of organisational learning. A post-mortem review is "a collective learning activity which can be organised for projects either when they end a phase or are terminated" (p. 295). They are used to reflect what happened in the past in order to improve future practice. Birk Dingsøyr, and Stålhane (2002, p. 43) state that post-mortem reviews ensure that teams remember and recognise what they have learned in a project. The same sentiments are shared by Collier, DeMarco and Fearey (1996), who state that the main reason for conducting a post-mortem review is to learn from the past. Desouza, Dingsøyr and Awazu (2005, p. 203) state that conducting a post-mortem is salient in order to gauge what has been learnt, what were the challenges and how they can be improved in future.

In their experimental use of lightweight post-mortem reviews, Birk *et al.* (2002, p. 45) revealed that a post-mortem review "is suitable when a project reaches a milestone and when the company is looking for qualitative experience that will help improve a similar, future project". They further state that a review is an excellent knowledge management technique that can improve software development activities if properly applied. Dingsøyr, Moe and Nytrø (2001) suggest a lightweight method to capture project experiences. They state that such a process is suitable for companies to learn from their mistakes. Their study found that this method captures more experience related to implementation, administration, developers and maintenance. Dingsøyr (2005) presents many ways of conducting post-mortem reviews in software organisations. These methods include those also suggested by Neal Whitten, Collison and Parcell and Birk *et al*. His study concluded by suggesting to companies that they must choose a review that is suitable to their needs. Ahonen and Savolainen (2010) conducted post-mortem reviews for five software development projects to investigate why the projects failed. They state that the review indicated that failure could start at the beginning of the project or during the project. They further state that projects fail because of unrealistic goals, customers being bankrupt, and staffing decisions, among others. Anquetil *et*

al. (2007) conducted experiments to investigate knowledge capture from legacy software maintenance projects. Their study was based on the fact that during software maintenance a lot of knowledge is lost; what remains is held by individual developers. They conducted experimental post-mortem analysis to capture and store such knowledge. The results of their experiments indicate that: 1) reviews are capable of uncovering lessons from past projects, 2) other types of knowledge (application domain knowledge) were obtained from such analysis, 3) the experiments helped create a culture of post-mortem analysis in the research organisations (p. 528). Bjørnson, Wang, and Arisholm (2009) also conducted an experiment to test whether a new proposed post-mortem review framework was more effective than the original framework. Their experiment revealed that the proposed framework worked better than the original one. Desouza, Dingsøyr and Awazu (2005) conducted a study to investigate how reviews could be used to capture tacit knowledge in a project. Their paper gives a detailed explanation of how the reviews benefit individuals, teams and the organisation. It also discusses the different post-mortem methods that organisations could use. Their study also revealed that such reviews could be done as reports or stories. They concluded that, if done properly, post-mortem reviews can help an organisation conduct effective and efficient project management (p. 214).

The literature on learning in software organisations indicates that learning starts from individuals, spreads to teams and then to the organisation. Unfortunately, not many studies have been conducted to investigate learning by individuals. It is still not clear how individuals learn (that is, what are the individual learning processes?). It is also not clear under what conditions individuals learn, and what tools facilitate individual learning. The literature covers a few studies about team learning. It reveals team learning activities, tools used by teams to learn, the conditions under which teams learn and team learning challenges. The literature reports extensively on how software organisations learn: organisational

learning processes, conditions for learning, tools that support learning, the benefits of learning and the use of knowledge management techniques to support learning. The literature also reports extensively how software organisations use post-mortem reviews to learn from past mistakes. What is lacking in the literature is a detailed description of how the knowledge management processes that support learning at individual and team level work and how the learning of individuals and teams enables learning at organisational level. There is also a lack of knowledge on whether knowledge management and organisational learning have reduced software development failures.

3.4. General knowledge management practices in software organisations

This section reviews literature on general knowledge management practices. By general practices we mean a combination of all the processes of knowledge management. The focus is not on one aspect only, for example knowledge creation or transfer, but on all processes. Studies under this section can be grouped into three broad categories: knowledge management in the software development process, knowledge management practices of people (individuals and teams), and knowledge management in the organisation. Knowledge management in the organisation includes software development processes.

3.4.1. Knowledge management practices in software organisations

The knowledge management literature also reports on general knowledge management activities in software organisations. This is research that investigates knowledge management practices in small and medium organisations, or knowledge management practices in general in organisations. Examples are Mathiassen and Pourkomeylian (2003) who report on the relationship between SPI and knowledge management and how this relationship unfolds in software organisations. Mathiassen and Pourkomeylian (2003, p.74) state that SPI activities support knowledge management by improving network facilities to create and store knowledge. They further state that SPI activities support knowledge sharing through informal

meetings, conferences, email and telephone. They conclude by stating that SPI is a particular form of knowledge creation and sharing. Mukherji (2004) reports on knowledge management practices in software service organisations. Mukherji (2004, p.35) states that software organisations create knowledge repositories of experiences for knowledge management purposes. These knowledge repositories help organisations store and transfer know-how, building communities of learning and sharing best practices. The study also revealed that software organisations also use expert locators to locate knowledge within the organisation and virtual spaces for sharing and creating knowledge within communities of practice in the organisation.

Feher and Gabor (2006) investigated possible knowledge management practices in software organisations and their possible support factors. Their study revealed that software companies applied both the codification (knowledge repositories) and personalisation (expert systems, data analysing tools, and knowledge maps) strategies for knowledge management purposes. They state that most companies are engaged in knowledge leveraging and transferring existing knowledge and creating and developing new knowledge and acquiring and integrating knowledge from external sources (p. 255).

Mehta (2008) conducted a study that sought to establish how globally dispersed software development organisations with successful knowledge management initiatives manage their existing knowledge resources, and at the same time improve existing ones and develop new for value creation. Their study discovered that successful knowledge management programs in global companies are influenced by technological, cultural strategic and procedural issues. They also say that in such environments, knowledge sharing is visible, transparent and traceable. Basri and O'Connor (2011) investigated knowledge management in small software companies. Their aim was to determine the role knowledge management plays in software development companies, that is, how software development knowledge is managed. Their

study revealed that in small software organisations knowledge is managed through continuous communication of teams, the sharing of knowledge by teams and team learning from knowledge. However, they discovered that small organisations do not document their knowledge well. Heavin and Adam (2012) conducted a similar study to investigate knowledge management activities in small and medium software companies. They discovered six knowledge management activities in small software development companies. They are: knowledge acquisition, codification, storage, maintenance, transfer, and creation. They concluded that, depending on the nature of the organisation, some activities are performed more than others.

Sharma, Singh and Goyal (2012) also investigated knowledge management practices in small software organisations. They found that small software organisations in India rarely use knowledge repositories to store their knowledge. The study also discovered that software developers prefer external sources of knowledge such as the Internet to create knowledge and that they sometimes use technologies such as email and groupware, corporate Internets/Extranets, data mining tools, and document and content management systems to manage their knowledge. Abdullah, Eri and Talib's (2011) study shows the importance of knowledge management and SPI in software organisations. The study concluded that knowledge management is very important in SPI for the facilitation of knowledge use within a software organisation.

Raju (2012) reports on the role of knowledge management in software organisations. Raju gives reasons why knowledge management is important in software organisations. The reasons are that the software industry has global distributed software development teams, it has iterative-incremental development, quick testing cycles, mergers and acquisitions, applies software outsourcing models, and the size of the organisations need knowledge management.

Other studies have focused on knowledge management activities in software organisations in specific countries or parts of the world. These studies are similar to the ones discussed above with only one noticeable difference: the focus is on organisations in certain countries. A few examples are Aurum, Daneshgar, and Ward's (2008) study that focused on knowledge management initiatives in Australian software organisations. The study found the following knowledge management practices: identification (re-using past knowledge to identify current problems), knowledge distribution (sharing knowledge with colleagues), acquisition, adaptation, creation (sharing ideas with colleagues), organisation (directory), and application (developers re-applying their accumulated knowledge in a rage of scenarios). They also use information networks of colleagues to share knowledge, and that they store their knowledge in repositories, directories, and archives (p. 523).

In Ireland, Heavin and Adam (2012) found that small software development organisations engage in six knowledge management activities. They are knowledge codification and acquisition, creation, storage, transfer, and maintenance. Pástor, Šipikal and Rehák (2013) investigated knowledge creation and acquisition activities in Slovakian software organisations. They found evidence of knowledge creation through the innovation and production of their own products and that knowledge is acquired from external sources. Neves *et al.* (2014) wanted to find out the role of knowledge management in mitigating risk in small software organisations in Brazil and found that combination (as defined by Nonaka & Takeuchi, 1995) is highly effective in mitigating risk in software development teams. However, they state that internalisation could also be used as well. Betz, Oberweis and Stephan (2014) investigated knowledge transfer challenges in off-shore software development focusing on German organisations. Their study identified 18 challenges spread across the eight knowledge management processes they adopted. They also identified 18 solutions to the knowledge transfer challenges.

The literature under this section discusses four issues, knowledge management practices in software development organisations, the importance of knowledge management in software organisations, knowledge management tools used in software organisations and knowledge management practices in software organisations in specific countries. One noticeable gap in the literature is the lack of studies conducted in Africa, especially South Africa. This study aims to fill that gap.

3.4.2. Knowledge management practices in the software development process

Knowledge management practices in the software development process are also reported in the literature. This is how knowledge management is applied in the actual development of the software. According to Dorairaj, Noble and Malik (2012, p.1), software development is a series of knowledge intensive tasks that includes requirements gathering, problem analysis, designing, coding, testing and maintaining the system to make sure it is updated and bug-free. Depending on the software development methodology being used, software development processes include planning and selection, systems analysis, systems design, and systems implementation and operations (Zhang *et al.,* 2005, p.515). Jiang *et al.,* (2009, p.1) talks about a System Infrastructure Development Life Cycle (SIDLC) which consists of six processes: requirement gathering, analysis, design, testing, implementation, and maintenance. Knowledge management could be applied in any of these steps. Studies in this category are classified into four categories: architectural knowledge management, the role of knowledge management in mitigating risk in software engineering, knowledge management in software testing, and knowledge management in the general software development process.

3.4.3. Architectural knowledge management

Clerc, Lago and van Vliet (2011, p.1) assert that architectural knowledge is an integrated representation of the software architecture of a software-intensive system, the architectural environment and architectural design. Architectural knowledge management is said to address challenges innate in global software development teams (Clerc, Lago & Vliet, 2011). In a literature review of architectural knowledge management in global software development teams, Beecham and Mistrik (2010, p. 350) identified three knowledge management strategies in global teams: knowledge codification, personalisation and a hybrid of the two. They state that codification involves storing architectural knowledge in a repository; personalisation involves individuals storing knowledge themselves [in their heads] and encouraging others to seek that knowledge from knowledgeable individuals. The hybrid strategy involves creating a repository and allowing people to use the repository to store and share their personalised knowledge.

Clerc, Lago and Vliet (2007) found two knowledge management practices in global software development teams engaged in architectural knowledge management. They found that teams document their knowledge and use different channels to share that knowledge. They indicated that teams share knowledge through face-to-face interaction and through technology (organisational portals, email systems, chat services and Wikis). In their later study, Clerc, Lago and Vliet (2011) found that global agile teams manage architectural knowledge by sharing it across sites through face-to-face meetings and using technology, writing down architectural rules, installing a directory of experts, creating a repository of architectural artefacts. Farenhorst *et al.* (2008, p.128) also report on a knowledge sharing technology used by software architects. They state that knowledge is shared via a knowledge repository, an expert's site, a knowledge maps system and a file share system.

3.4.4. Knowledge management in mitigating risk in software development

Other studies focus on how knowledge management practices could be applied to reduce risk in the software development process. Facin, Laurindo and Spinola (2014) state that risk is one of the factors that cause software development failures. They conducted a study to investigate how knowledge management can reduce risk in software development. Their study revealed that organisations could employ knowledge management tools such as portals to capture, store, and share risk knowledge. They concluded that knowledge management helped organisations collect knowledge about risks associated with the software development process creating conditions to reuse such knowledge in future projects. Neves *et al.* (2014) investigated knowledge management practices that could minimise risk in software development. They found three major knowledge management activities to analyse risk: conducting meetings and training sessions and creating and using knowledge repositories (Neves *et al*., p. 134).

Chang (2013) proposed a knowledge management mitigating risk framework for software development. Chang (2013) is of the view that storing, mining and using knowledge from past projects can mitigate risk in software development. Alhawari *et al.* (2012) proposed a knowledge based risk management framework that illustrates the role of knowledge management processes in enhancing and facilitating risk identification, analysis, risk response planning and execution processes in IT projects. The framework suggested how knowledge management processes could be integrated into core risk management processes to mitigate risk in IT projects.

3.4.5. Software testing and knowledge management

Knowledge management initiatives have also been studied in the sub-process of software testing. De Souza, Falbo and Vijaykumar (2014, p.2) define software testing as a process of the verification and validation of the behaviour of a program against the expected behaviour.

They are of the opinion that in the field of software testing, knowledge management "can be used to capture knowledge and experience generated during the testing process". In their literature review of knowledge management practices in software testing, de Souza, Falbo and Vijaykumar (2014) state that knowledge management studies in this sub-field focus mainly on general knowledge management practices such as knowledge representation, capture and retrieval. They further state that the literature indicates that knowledge management practices are employed in software testing to encourage software reuse and promote organisational learning. Andrade *et al.* (2013, p. 20) give four reasons why knowledge management is important in software testing. They state that it is important because it improves processes, it retains knowledge in corporate memory, it institutionalises best practices, and helps to overcome challenges set by new methodologies and techniques.

Karhu, Taipale, and Smolander (2009) investigated knowledge transfer in software testing. Their study revealed an increase in knowledge transfer between testing and earlier stages of the software development process and that the knowledge transfer is associated with testing schedule overruns. Nogeste and Walker (2006) conducted an action learning case study to develop and implement a process improvement based on a knowledge management framework in software testing. Their framework provides a simple and effective way of tapping into a knowledge pool. Nogeste and Walker (2006) concluded by encouraging project managers, team leaders and line managers to consider using the framework as a means of defining knowledge management processes for project work and the day-to-day running of the organisation. Liu, Wu, Lui and Gu (2009) investigated knowledge management practices in software testing and technologies used in such environments. Their study revealed four major knowledge management activities in software testing, which are ontology based knowledge representation, knowledge transfer, constructing a knowledge map, and ontology-based searching and retrieval activities. Their study constructs a knowledge management

system model to be used in software testing environments. Abdullah, Eri and Talib (2011) developed a knowledge management system model for a community of practice in a software testing environment. Their model is based on the knowledge management life-cycle (create, capture, refine, store, manage, and disseminate). They conclude that this model is a very important feature for the community of practice to manage knowledge in the software testing environment.

3.4.6. Knowledge management in the general software development process

The role of knowledge management in the software development process is reported extensively in the literature. This section discusses how knowledge management is used in software development processes and how different software development methodologies support knowledge management. The literature indicates that studies in this category focus on three areas: knowledge management practices in the systems development life-cycle (SDLC), knowledge management in global software development, and knowledge management tools used in the software development process. Kukko, Helander and Virtanen (2008) investigated knowledge management challenges in software development process renewal. They found two broad challenges, which are technology oriented challenges and human oriented challenges at the design and implementation phases of software development (p.5).

Technology oriented challenges have to do with teams and demands, lack of time for training and experimenting, usability and exploitability of the component library, finding a viable common technology that meets the needs of different teams, the fit between the initial and new technology. Human oriented challenges include among others, prejudices towards new technology and attitude towards a new technology. Jelínek's (2010) study discusses how knowledge management is applied in different software development methodologies. Jelínek discussed the application of knowledge management in three software development methodologies: the waterfall, agile and Rational Unified Process (RUP). Jelínek uses the

SECI model to describe the knowledge management processes in each of the methods. Different stages of the methods are explained; how they support knowledge management and the knowledge management challenges they face. Lakalu (2010) formulated a knowledge management system framework for open-source software development. According to Lakalu (2010, p. 82) the framework enabled communities of practice to share the knowledge of open-source development. The framework proposes mechanisms for knowledge sharing throughout the software development process. It gives opportunities and guidelines for communities of practice to share their knowledge and encourage people to adopt the open-source software (OSS) methodology in software development throughout the development phase.

Knowledge management in offshore software development seems to be gaining popularity in the software development field. A lot of focus has been given to this area recently. Some software development journals have dedicated issues on this topic. For example, the *Expert Systems Journal* dedicated its July 2014 issue to this topic. Mathrani, Parson, and Mathrani (2012) investigated how offshore software development (OSD) vendor organisations in different global environments manage knowledge based activities in offshore software development projects to be successful.

Mathrani *et al*. (2012) found knowledge management initiatives at two levels of software development, which are the operational and design levels. Knowledge management activities such as knowledge dissemination and skills development are common at operational level, and phenomena and activities such as staff attrition, project management and quality assurance are common at design level. Kristjansson, Helms, and Brinkkemper (2014) investigated knowledge sharing in offshore software development. Their study was motivated by the knowledge sharing challenges that software organisations face when they outsource their development tasks. Their study provides a normative knowledge transfer model that

addresses three main phases (team handover, project kick-off and on-going development) in software-outsourcing projects and contains six best practices (knowledge alignment, functional collaboration, specification grooming, technical assistance, technical documentation, and product transition) and a blueprint for a knowledge infrastructure to support knowledge exchange in the development phase of the project (p. 267). The model recognises that knowledge needs to be shared at different stages of the software development project.

Dingsøyr and Smite (2014) discuss knowledge management activities in global software development (GSD). Their study reveals three knowledge management activities. They state that GSD organisations use knowledge repositories such as corporate databases and intranets to store organisational knowledge. These organisations also use Wikis for knowledge mapping purposes, and use task boards to share and disseminate knowledge. Their study concluded that managing knowledge can be difficult in global software development. They suggest that organisations engaged in such activities must adopt different knowledge management approaches instead of one, to define what knowledge is to be shared locally and globally, and adopt local knowledge management approaches that require fewer resources. Parviainen and Tihinen (2014) conducted a literature review on knowledge management challenges faced by global software development teams. They clustered the challenges based on surveys and industrial cases. Solutions to these challenges were later proposed.

Other studies focus on knowledge management tools that are used in the software development process. For example, Kamunya and Waweru (2013) present different tools that can be used by software organisations to manage their knowledge. They state that such tools make it possible for knowledge to be available to developers, thus helping them in the software development process. They list tools such as content management systems,

knowledge sharing tools and knowledge search and retrieval tools (p. 4) as essential tools in the software organisation. Portillo-Rodríguez *et al.* (2012, p. 671) found that organisations use tools such as Wikis, document management systems and blogs for knowledge management purposes in distributed teams. Menolli *et al.* (2015. P. 296) provide a list of knowledge management technologies used by software organisations in Brazil. They are Wikis, collaborative writing tools, social networks, repositories of shared documents, and property tools. They state that Wikis and document repositories are mostly used for knowledge storage and knowledge transfer respectively; blogs and collaborative writing tools are used for knowledge sharing and reuse.

3.4.7. Knowledge management in software development teams

The knowledge management literature also covers the role of knowledge management in software development teams. Such studies discuss how knowledge management improves team performance. Bowen and Maurer (2002) present a case of a knowledge management system used by a distributed software development team. They state that the system enabled the team to maintain adaptive practices in a distributed setting, supports information routing, project coordination, pair programming, team communication, and experience management (p. 1). Taweel, Delaney and Zhao (2009) conducted a similar study to Bowen and Maurer's. Their study investigated the knowledge needs of distributed software teams. They found that distributed teams face challenges about project knowledge. Dorairaj, Noble and Malik (2012) investigated knowledge management practices in agile distributed teams. Their study describes how agile teams gather, store, share and use knowledge. Dorairaj, Noble and Malik (2012, p. 7-12) describe in detail knowledge management practices in agile teams. They state that agile teams generate knowledge through project inception practices (brainstorming about the project), customer collaboration, formal training, communities of practice, and self-learning. Agile teams codify knowledge by using Wikis, documents, and technical

presentations. Knowledge is transferred through daily scrum meetings, project inception, pair programming, using tools such as the Internet, team visits, team rotations, on-site customer input, cross functional teams and discussions. Knowledge gained in the past is applied in similar current contexts by solving current problems. Mathrani, Parsons and Mathrani (2012) investigated knowledge management practices in offshore software development. They chose the offshore business model because it is the dominant software model these days because of low development costs. Their study revealed that software development companies value the need to create knowledge and disseminate it within the organisation. They also revealed that knowledge management initiatives occur at two levels in the organisations: design and operational levels. Knowledge management initiatives found at operational level are knowledge sharing through face-to-face interaction and social networks (p.2717). Knowledge management activities at design level involve the development of and access to explicit knowledge in knowledge repositories (p, 2720).

The literature on general knowledge management discusses studies that are not specific to a certain process but focus on general knowledge management processes in many situations. The literature discusses how knowledge management practices are applied in software teams, in software organisations, to reduce software development risk, in software testing, in architectural knowledge management and in the actual building of the software. What could be deduced from these studies is that they focus mainly on the knowledge management life-cycle in different contexts. That is how teams and organisations acquire, create, store, share, use, organise and learn from knowledge.

3.5. Software developers' knowledge management practices

The literature is quite scarce on software developers' knowledge management practices. A few studies are found on this issue. Rezende and Alves (n.d.) interviewed software developers in Brazil to find their knowledge management practices. Their study found four generally adopted knowledge management practices: creating and updating databases of best practices and lessons learned and listing of experts; preparing written documents such as manuals, lessons learned, and articles for publication; and using the Internet to acquire knowledge. Sharma, Singh, and Goyal (2012) found that developers are aware of the knowledge management concept and that they acquire knowledge from internal and external sources. Internally, they acquire knowledge from colleagues; externally from the Internet. Their results further indicate that developers reuse their knowledge. They then indicated challenges that hinder the full adoption of knowledge management activities. They mentioned the lack of monetary and other incentives as the main reason.

There is a severe scarcity of literature on software developers' knowledge management practices globally and also specifically in the South African context. Very few studies were found addressing this issue. This study tried to fill that gap as well and investigated the knowledge management practices of developers. This is because, without the developers adopting knowledge management practices, it is difficult for the whole organisation to adopt them.

3.6. Knowledge Management challenges

In as much as knowledge management has been adopted and promises to help organisation improve their routines and processes, it has its own challenges. These challenges are caused by a number of factors. Herrmann (2011, n.p.) explains a number of challenges that could hinder the implementation of knowledge management in organisations. They organisational, technology, content, routines and procedures, and personnel challenges. According to

Herrmann, organisational challenges have to do with the knowledge-sharing culture of the organisations. Technology challenges are related with the difficulty in accessing and using technology. Content challenges have to do with collecting and storing organisational knowledge. Routines and procedures challenges are associated with tasks that are not recognised by employees or management and personnel challenges have to do with individual employee's behaviour. Raths (2012) mentions sharing of best practices as the main barrier to knowledge management adoption. Gupta, Iyer and Aronson (2000, p.19) are of the view that the main challenge facing knowledge management implementation is the difficulty to encourage, coerce or direct people to share knowledge.

Soakell-Ho and Myers (2011, p. 212) state that organisational structure and culture are the main barriers to knowledge management implementation. Organisational structure issues include among others the centralisation and decentralisation of tasks within the organisation. Organisational culture has to do with different cultural aspects of the employees. Soakell-Ho and Myers (2011) also found other challenges that are associated with technology (accessibility and use of IT and its role in knowledge management), lack of funding initiatives for knowledge management, and external relationships. Technological challenges are also raised by Kiniti and Standing (2013) who state that technology can fail knowledge management if it is not adopted. Kalkan (2008, p. 394) also indicted that IT is a knowledge management barrier if it is poorly implemented.

Kalkan (2008) also found that dealing with tacit knowledge, defining knowledge, cultural complexities, organisational structure, human resources, and coping with competition are major knowledge management barriers. Chen, Tong and Ngai (2007, p. 138) state that trust and conflict, communication mechanisms, knowledge transfer and organisational structure are the main knowledge management challenges.

Software organisations too, do experience knowledge management challenges. The main challenge that they face is knowledge sharing, especially in development teams. Betz, Oberweis and Stephan (2014) identified 18 challenges spread across eight knowledge management processes. Kukko, Helander and Virtanen (2008, p. 5) found two broad challenges, which are technology and human oriented at the design and implementation phases of software development. Technology oriented challenges have to do with teams and demands, lack of time for training and experimenting, usability and exploitability of the system, finding a viable common technology that meets the needs of different teams, the fit between legacy systems and new technology. Human oriented challenges include among others, prejudices towards new technology and attitude towards a new technology.

Jelínek's (2010, p. 36) found that the biggest knowledge management challenge in software development is sharing tacit knowledge especially in the waterfall method. Kristjansson, Helms, and Brinkkemper (2014, p. 267) found that knowledge transfer is a key challenge in outsourced development work.

The literature indicates that organisations in general, whether software development or not face knowledge management challenges. These challenges are associated mainly with organisational structure and culture, technology and the nature of knowledge (tacit knowledge).

3.7. Summary

The chapter presented a review of the existing literature on software development failures and their intervention strategies. It also presented the literature on knowledge management practices, benefits and challenges in the software engineering environment. The literature revealed that software development project failure is still prevalent in software development organisations globally and also in South Africa. The failure rates are very high. The failures are caused by a number of factors including unarticulated requirements, failure to learn, lack of management support and many other factors. The literature indicated that there are intervention strategies (SPI and agile methods), but they don't seem to work because the failures are still reported. The literature also reports that the software engineering field has adopted knowledge management practices. Studies have been conducted in different sub-fields of software engineering. For example, in software testing, architecture, risk, software development teams and generally in organisations in different parts of the world. The studies adopt the lifecycle framework. That is, they mainly regard knowledge management as a series of processes such as knowledge acquisition, creation, sharing and many other processes.

However, the literature reveals a scarcity of such studies in South Africa. The very few studies that are found in South Africa do not focus on knowledge management practices from the whole lifecycle perspective, but focus mainly on one or two narrow aspects of the lifecycle, for example, a particular study focusing only on knowledge sharing in software organisations. There is also a shortage of studies that focus on individual software developers' knowledge management practices. This is happening globally. The focus is mainly on teams, especially distributed teams. This study addressed these concerns by investigating knowledge management practices in a South African software development environment focusing on individual software developers and the whole organisation. The

study also adopted a unified knowledge management framework instead of focusing on a narrow area of the framework.

CHAPTER FOUR
RESEARCH METHODOLOGY

4.1. Introduction

In every study, the researcher must clearly state the methods used to conduct it. This chapter discusses the research design of this project. It discusses the philosophical underpinnings, that is, the ontological and epistemological perspectives. It goes on to discuss the research methodology and subsequent methods adopted. It also discusses the sampling and data collection procedures, ethical considerations, issues of validity and reliability and the logic of reasoning behind the study. The chapter is structured as follows: the philosophical underpinnings are discussed first. They are followed by the logic of reasoning and the research methodology, followed by data collection procedure. This includes the population of the study, sampling procedure, and research instruments. Ethical issues are discussed after the data collection procedure, followed by the last two sections, validity and reliability and time horizons.

4.2. Philosophical perspectives

Social research is informed by philosophy, which is used by researchers to inform their studies. There are ontological and epistemological philosophies that inform every research project. This section discusses the philosophical underpinnings of this study.

4.2.1. Ontological perspective: relativism

Schuh and Barab (2008) define ontology as the real world, whether physical or abstract structures or what exists. Marsh and Furlong (2002) calls it a theory of being. Two ontological assumptions are widely accepted in research: realism and relativism. Realism is a view that supports the existence of a real, physical world independent of individuals and human experience. This view assumes that the world exists separately from human

perception and mind (Schuh & Barab, 2008). Relativism assumes that reality exists through social interactions and experience with the environment. It posits that reality is constructed socially through social interaction with the environment. Although the phenomena under investigation could be investigated with any ontological stance, this study has adopted the relativist ontology. It views knowledge and its management as a socially constructed phenomenon with the human being deeply involved in its formulation. In this study, knowledge management is regarded as a social phenomenon which comes into being through experience by socially interacting with the environment. In organisations, knowledge management occurs when individuals socially acquire, create, organise, share and apply knowledge. This study views organisations as collections of individuals and teams that are socially connected.

4.2.2. Epistemological perspectives: Interpretive paradigm

Burrell and Morgan (1979, p.1) define epistemology as assumptions that ground knowledge, that is, how one understands the world and communicates the world to other people. These assumptions entail ideas about how knowledge is obtained or what is true or false knowledge. Epistemology in research is explained by research paradigms. It is a set of assumptions adopted by a community of practice that allows its members that share similar perceptions to engage in commonly shared practices. It is about how knowledge is acquired in the physical and social world (Hirschheim & Klein, 1989, p. 1201).

Burrell and Morgan (1979, p. 23) describe four research paradigms that are applied in social research. They are the radical humanist, radical structuralist, interpretive, and functionalist paradigms. They state that functionalism is rooted from the objectivist point of view. It aims to explain the status quo, social order, consensus, social integration, actuality, solidarity, and need satisfaction. It seeks to provide rational explanations of social affairs and is concerned with the generation of knowledge that can be used. Its main aim is to “understand

relationships between objective social facts and to articulate the sociology which explains the types of solidarity providing social cement which holds society together" (p, 26). The interpretive paradigm is concerned with understanding the world as it is through subjective experience. It sees the social world as a social process which is created by human beings (Burrell & Morgan, 1979, p. 28). Radical humanism is concerned with radical changes from a subjectivist standpoint. It is closely related to interpretivism. It is concerned with over-throwing the limitations of existing social arrangements and criticising the status quo (Burrell & Morgan (1979, p. 32). The radical structuralists advocate radical change from an objectivist standpoint. The Burrell and Morgan framework has been adopted by researchers such as Chua (1986, p. 603), who also identified four research paradigms: radical structuralist, interpretive, the functionalist, and radical humanist. It was also adopted by Hirschheim and Klein (1989, p. 1201) who describe four research paradigms, which are functionalism, neo-humanism social relativism, and radical structuralism.

Chua (1986) classified research epistemologies into positivism, interpretivism, and critical. Olikowsky and Baroudi (1991, p. 5-6) adopted Chua's framework and classified research philosophy into the positivist, interpretive, and critical paradigms. They state that the main purpose of positivistic research is to test theory by formulating formal hypotheses, measuring quantifiable variables, testing the hypotheses, and the drawing of inferences about phenomena from a sampled population. They also state that interpretivism assumes that "people create and associate their own subjective and inter-subjective meanings as they interact with the world around them" (p. 5). Critical studies aim to critique the status quo and change what is believed to be not right in society.

Schultzer and Leidner (2002) adopted Deez's (1996) framework and identified four scientific discourses to study knowledge management in organisations. They are the normative, the critical, the interpretive, and the dialogic discourses (Schultzer & Leidner, 2002, p. 213). The

focus of the normative is the discovery of technology solutions. The interpretive discourse focuses on the role of knowledge in organisational transformation and the role of technologies in supporting knowledge work. The research focus of the critical discourse is on power relations in organisations. The knowledge metaphor is knowledge as a commodity. The focus of the dialogic discourse is the dynamically intertwined and conflicting nature of organisational memory and organisational forgetting, as well as the implications of forgetting and memory and on visibility, identity, and power (Schultzer & Leidner, 2002, p. 217).

The study adopted the interpretive paradigm as defined by Burrell and Morgan (1979) and Olikowsky and Baroudi (1991). From the knowledge management perspective, the study adopted Schultzer and Leidner's (2002) definition of interpretive research. This is because the phenomenon under investigation (knowledge management) is a subjective concept. It is human beings interacting with each other and knowledge objects that can lead to knowledge management. Knowledge management is a social process which is dependent on human socialisation. The study interpreted the actions of actors to determine knowledge management practices. The next section discusses the logic of reasoning which the study adopted.

4.3. Research methodology

Fassinger and Morrow (2013, p. 74) are of the view that the choice of a relevant method in a study is determined by the research question(s) to be answered. They are supported by Venkatesh, Brown and Bala (2013) who state that the choice of methods depends on the research purpose, context and question. This section discusses the research methodology adopted in this study. The section briefly discusses different research methodologies used in social research and then discusses in detail the methodology of choice. First to be discussed is quantitative methodology, followed by qualitative methodology and then multiple-methods.

4.3.1. Quantitative methodology

Research in the social sciences is usually quantitative, qualitative or a combination of the two methods. Creswell (2009, p. 233) defines quantitative research as a means of testing objective theories by looking at the relationship between variables. Creswell (2009) further states that variables can be analysed using statistical procedures after being measured using different instruments. Yilmaz (2013, p. 311) defines quantitative research as concerned with investigating phenomena according to numerical data which are analysed mathematically. It is empirical research investigating social phenomena, testing theories consisting of variables which are measured with numbers and analysed statistically to determine causal relationships among them. Quantitative studies are concerned with generalisations, outcomes, and cause-effect relationships through deductive reasoning and prediction. They are suitable methods when a large population is to be studied with a few questions and the results are generalisable to that population (Yilmaz, 2013, p. 313). Examples of quantitative methods are surveys and experiments.

4.3.2. Qualitative methodology

Qualitative research is defined by Creswell (2009, p.232) as a method of exploring the meaning of groups or individuals related to a social or human problem. Myers (2013, p. 5) defines qualitative research as "designed to help people understand what they do". They help researchers understand social phenomena in its natural context. "Qualitative studies are concerned with processes, context, interpretation, meaning or understanding through inductive reasoning" (Yilmaz, 2013, p. 313). Examples of qualitative research methods are ethnography, action research, case study research, grounded theory and archival research. Qualitative data sources include interviews, participant observation, documents and texts and the researcher's impression of situations (Myers, 2013). This method is best when a

phenomenon is to be studied in-depth. It is also good for exploration where there is no theory but theory could be developed and tested by quantitative methods (Myers, 2013, p. 9).

This study adopted a qualitative approach, which was suitable because of three main reasons: the population was very small, no associations and relationships were tested and the main research question was seeking explanations from respondents and those responses were to be interpreted. The research question entailed interviews being conducted with respondents.

4.3.3. Mixed/Multiple methodology

According to Venkatesh *et al.* (2013, p. 23), mixed methods is a research method that uses multiple methods, that is, more than one method and more than one world view occur in a research inquiry. Greene, Caracell and Graham (1989, p. 256) define mixed methods design as one quantitative method and one qualitative method, where neither type of method is inherently linked to any paradigm of inquiry. Johnson and Onwuegbuzie (2004, p. 17) define mixed methods as a situation in which quantitative and qualitative concepts, techniques, methods, approaches and languages are mixed or combined in a single study in order to legitimise the use of multiple approaches to answer a research question rather than constraining the researcher to one choice.

Creswell (2009, p. 230) defines mixed methods research as an approach that combines both qualitative and quantitative forms of research in a single study. There are three types of mixed methods research described by Creswell (2009, p. 14-15), sequential, concurrent, and transformative. Sequential mixed methods elaborate the findings of one method with another method. Concurrent mixed methods entail the researcher using both methods in the same study concurrently. Transformative mixed methods are those in which the researcher uses a theoretical lens as a perspective within a study that combines quantitative and qualitative data. Blaikie (2010, p. 219) states that using mixed methods encourages researchers with

different skills and views to collaborate, provides more comprehensive evidence; the strength of one method offsets weaknesses in other methods and it also encourages the use of multiple paradigms.

Mixed methods are closely associated with multiple methods. This has led to confusion of the two concepts as stated by Venkatesh *et al.* (2013, p. 23). They state that multiple research methods are sometimes confused with mixed methods research as if these concepts mean the same thing, yet there are significant conceptual differences between the two concepts. The following section discusses the research method used in the study.

4.4. Research method: Multiple Case study method

There are many research methods that the social researcher can choose from in a research project. It all depends on the question(s) to be answered. They can choose from quantitative methods such as surveys and experiments or qualitative methods such as case studies, ethnography, action research, narrative analysis and many others.

This study adopted the multiple case study research method. Lee (1989, p. 34) defines a case study as an examination of a real-world phenomenon as it exists in its natural settings. Palvia *et al.* (2003, p.292) define a case study as a study which studies a single phenomenon in-depth in its natural setting. That setting could be a single or multiple organisations. The same sentiments are shared by Benbasat, Goldstein and Mead (1987, p. 370) but they define it further by adding that a case study employs multiple data collection methods from one or a few entities, that the boundaries of the phenomenon are not clear at the beginning and no experimental control or manipulation is used. Benbasat *et al.* (1987, p. 370) believe that the case study research allows the researcher to study the phenomenon in its natural settings, that it asks 'how' and 'why' questions, is suitable to study an area in which few studies have been conducted. Yin (1999, p 1211) also shares similar sentiments and states that "the all-

encompassing feature of a case study is its intense focus on a single phenomenon within its real-life context." Yin (2014, p.16) defines a case study in terms of its scope and its characteristics as a method of inquiry. According to Yin, (2014, p.16) a case study investigates a phenomenon in depth and within its natural settings more especially when the boundaries between phenomenon and context cannot be clearly defined.

Yin's definition of a case study is partly dismissed by Myers (2013). The latter states that Yin's definition does not fit all case studies, especially in the business discipline. Myers (2013) contends that Yin's definition can be too broad and too narrow at the same time. It is broad in the sense that case studies in business studies usually focus on one or more organisations, and it is too narrow because Yin's definitions advocate only one type of case study research. Myers assumes that this is because of the positivistic nature of Yin's definition. He states that such a definition is not suitable for all qualitative case studies. Myers (2013) then defines case study research as a research method that uses evidence from one or more organisations where the phenomenon is studied in context using multiple sources of evidence, although most of the evidence comes from interviews and documents (p. 78).

Scholars have agreed that case study research investigates the phenomenon in its natural settings, involves one or more cases (organisations, processes, people, etc.) and that data are collected from multiple sources.

This study adopted these conditions of a case study, but leans more towards Myers's definition. This case study investigated the phenomenon in its natural settings, namely software organisations in the province of KwaZulu-Natal, South Africa.

Yin (2014, p. 50) states that case study research can take two forms, i.e. a single case study and multiple case study. A multiple case study happens across sites (Hammond &

Wellington, 2013, p. 18). Yin (2009, p.20) states that multiple case studies draw a single set of cross-case conclusions.

Case studies, whether single or multiple, can be holistic or embedded. A single holistic case study investigates a single case with a single unit of analysis. A single embedded case study investigates one phenomenon and multiple units of analysis. The same description applies to multiple case studies. They can be holistic or embedded. This study was a multiple, embedded case study, because it investigated multiple organisations (12) and there were two units of analysis: individuals and documents.

Baškarada (2014, p. 3) and Myers (2013, p. 79) are of the opinion that case study research could be informed by any of the three research paradigms, namely positivism, interpretivism and critical paradigms. Such case study research projects follow their respective social research paradigms. This case study research followed the interpretive paradigm. According to Myers (2013, p. 80) interpretive case study research views social reality as socially constructed and attempts to understand phenomena through the meanings that are assigned by people and defines quality in terms of the plausibility of the story and the overall argument. It is not concerned with issues of validity and reliability, which are emphasised in positivist case studies. This is an interpretive case study.

As much as case-study research is popular in social research, Lee (1989, p. 35) describes four problems associated with the case study methodology; the difficulty in making observations and controlled deductions, and the difficulty in replication and generalisation of case studies. Yin (2014, p.19-21) discusses five case study concerns. They are rigour, confusion with teaching case studies, generalisability of case studies, the level of effort put in conducting a case study and the unclear comparative advantage. Myers (2014, p. 83) states that the disadvantages of case study research are the difficulty of gaining access to the research site,

the fact that the researcher has no control over the situation, the difficulty of carrying out the study especially for younger, inexperienced researchers to focus on important issues and the amount of time needed to conduct case study research.

This study was not immune to these problems, but the researcher was able to overcome them. Three issues stand out as mentioned by the above-mentioned scholars. There is no control over the research situation, there is difficulty in replication and generalisation, and a high level of effort in conducting a case study. Fortunately for the researcher, there were no disturbances that obstructed observation. The rigour, replication and generalisation problem has been addressed in the literature. Myers (2013) and Yin (2014) have addressed these issues. For example, Yin (2014, p. 21) explains that replication and rigour are addressed by performing multiple case studies and that generalisation of a case study is done to theoretical propositions and not the population. This research was a multiple case study inquiry. Issues of replication and rigour were addressed by the multiple cases conducted. Fortunately, this was a Doctoral study; the researcher had five years to finish the study so the issue of time was covered.

The multiple case study method has been used to study knowledge management in software organisations before. Boden *et al.* (2010) conducted multiple-case studies to investigate learning in software development companies. Heavin and Adam (2012) also conducted multiple case studies to investigate knowledge management practices in software organisations. Chouseinoglou *et al.* (2013) conducted multiple case studies to investigate knowledge management and learning in software organisations. Others were multiple case studies focusing on specific geographical areas. Examples are Pástor, Šipikal and Rehák (2013) who conducted a multiple case study investigating knowledge creation and acquisition activities in Slovakian software organisations, Betz, Oberweis and Stephan (2014) who investigated knowledge transfer challenges in off-shore software development focusing on

German organisations and Aurum, Daneshgar, and Ward's (2008) case study that focused on knowledge management practices in software organisations in Australia. This study followed the same research route as the studies mentioned above. The next section discusses how data were collected.

4.5. Population, sampling procedure

This section discusses data collection procedure. It discusses the population, the sampling procedure and the instruments used to collect data.

4.5.1. Population

The population of the study was software development organisations. This study defines a software development company or organisation as a company that specialises in the development of software products for clients. SMMEs in the KZN province were targeted. A SMME, depending on the sector of the South African economy, is an organisation with fewer than 200 full-time paid employees (The Department of Trade and Industry, n.d.). The Johannesburg Centre for Software Engineering Source Code Report (2012) states that most software organisations in South Africa are SMMEs. KZN was selected for two reasons; easy access to the research site, and the fact that KZN is the third largest host of IT projects in South Africa (FDI Intelligence, 2012). In the organisations, software project managers, software developers and software development related documents were targeted. Software development managers and developers were selected because they are directly involved in the software development process. Project managers responded to questions directed to the whole organisation, developers answered questions concerning individual practices.

4.5.2. Sampling procedure

A sample is a fraction of a population. A population is the universe of inquiry. It could be people, organisations, events or items that are relevant to the research problem (McGiven, 2006). Two types of sampling techniques are common in social research, namely probability sampling and non-probability sampling. In probability sampling, the elements have a known chance of being selected (Proctor, 2005). In non-probability sampling, the probability of elements being selected is not known, because the researcher might choose a particular element consciously or unconsciously (McGiven, 2006). Non-probability sampling techniques include judgemental/purposive, convenience, and quota sampling. Probability sampling techniques include simple random, stratified, systematic, and cluster sampling (Tustin, Ligthelm, Martins & van Syk, 2005).

This study applied non-probability sampling. Non-probability sampling was suitable because of the small target population and the unique nature of the organisations. The study specifically targeted software development organisations. Software organisations are unique in a sense that they are usually small in size and in number because they serve a niche market. Two non-probability sampling techniques were adopted. They are snowball and purposive sampling. The starting point of the sampling process was purposefully identifying software development SMMEs based in the province of KZN. A list of software development organisations in South Africa, provided by the Johannesburg Centre for Software Engineering Source Code Report, was used first. The list contained 12 organisations in the KZN province. At first, all 12 organisations were included in the sample. These organisations were then contacted telephonically to confirm their existence. This exercise confirmed the existence of only four organisations. During the confirmation exercise the researcher requested the companies to provide names of other organisations they knew in the area. Four organisations were identified. Web searches on the local business portal Snupit (http://www.snupit.co.za)

and the online Yellow Pages of South Africa website (http://www.yellowpages.co.za) were also conducted, and 21 organisations were found. The 21 organisations were confirmed to exist but only 12 agreed to participate. The total number of organisations that participated in the study is twelve. In each organisation, the software development project manager or IT manager and one software developer were targeted. One developer was chosen because it was realised during the pilot study that because of the small size of the organisations, some organisations had only one developer who also happened to be the IT manager. For consistency reasons, one developer was chosen in each organisation. In four of the organisations, the project manager was also the only developer. This means that the study was able to involve 12 project managers and eight developers. Table 8 provides a summarised profile of the organisations. For confidentiality purposes, they are called organisation A to L.

Table 8: Profile of the participating organisations

Organisation	**Profile**	**Size of the development team**
Organisation A	Was established 28 years ago and specialises in the development of point of sale software for the automotive sector.	2
Organisation B	Develops a range of solutions for the clothing and footwear industries.	3
Organisation C	This organisation develops business management, customer relationship management, image finder, questions and survey, and time and access software.	1
Organisation D	Develops software for the pharmaceutical industry for dispensing medication, point of sale and group management software.	4
Organisation E	Develops point of sale software for leading fast-food chains, for home delivery and table service. It has been in service for more than 20 years.	2
Organisation F	Was established in 2005. The company develops websites, eLearning software, and does graphics design.	2
Organisation G	Develops solutions for data storage, data retention and regulatory requirements.	3
Organisation H	It was founded in 2006 and focuses on website development and web applications.	1
Organisation I	Develops software for electrical network design that handles any voltage modelled in a network.	4
Organisation J	Specialises in data management services and training solutions.	4
Organisation K	Was established in 1983. It specialises in developing tax software for small and large businesses.	1
Organisation L	Develops websites and web applications.	1

The number of cases that are sampled in research varies considerably. It is acceptable to have a multiple case study with two or more cases to analyse. For example, Betz, Oberweis and Stephan (2014) sampled six German organisations in their study on knowledge transfer in offshore software development. Aurum, Daneshgar and Ward (2008) sampled only two organisations in Australia; Heavin and Adam (2013) sampled four software organisations in the Republic of Ireland and Pástor, Šipikal and Rehák (2013) sampled several companies in Slovenia. The current study involved 12 cases.

4.6. Validity and reliability

Validity and reliability are concepts that are closely associated with quantitative, positivistic research. It has been widely agreed that every study, whether quantitative or qualitative, must conform to validity and reliability standards. Quantitative and qualitative scholars have been debating what constitutes validity and reliability in social research. Quantitative scholars have however clearly defined validity and reliability in quantitative research and have generally adopted its guidelines. This is not the case in qualitative research. It is still not clear what constitutes validity and reliability in qualitative research. This section discusses the two concepts and their application in this qualitative research.

Validity is concerned with the accuracy of scientific studies (Le Compte & Goetz, 1982, p.32). A valid study must show what actually exists and a valid instrument must measure what it is supposed to measure. Bapir (2012, p. 15) defines validity in the context of quantitative research and states that it "concerns the relationship between the data and the construct, the findings and the conclusion, the reality and in the representation".

Le Compte and Goetz (1982, p.32) state that reliability is concerned with whether a study can be replicated. They further state that reliability requires that a researcher using the same methods can yield the same results as a previous study. Selltiz *et al.* (as cited in Brink 1993,

p. 35) defines reliability as "concerned with the consistency, stability and repeatability of the informant's accounts as well as the investigators' ability to collect and record information accurately".

Issues of validity and reliability in qualitative research have always been questioned by quantitative researchers (Le Compte & Goetz, 1982). This is because qualitative research has often been criticised for lacking scientific rigour and for a lack of transparency in the analytical procedure, poor justification of methods employed and findings being a mere collection of opinions (Noble & Smith, 2015, p. 34). This issue has caused a lot of debate in the social sciences, especially in qualitative research. Rolfe (2006, p. 304) states that three schools of thought have emerged from this debate. There are those who believe that the issue of validity and reliability must be treated the same way as in quantitative research; there are those who believe in a new set of guidelines in qualitative research and those who question the appropriateness of any predetermined criteria for judging qualitative research.

Other scholars such as Sandelowski have totally rejected the notion of validity and reliability in qualitative research. But scholars such as Morse *et al.* (2002) argue that validity and reliability is relevant in qualitative research. They urge qualitative researchers to implement integral and self-verification strategies in any inquiry. Qualitative researchers tend to avoid the concepts validity and reliability, but prefer concepts such as trustworthiness (Guba & Lincoln as cited in Morse *et al*., 2002), credibility and transferability (Lincoln & Guba as cited in Bapir, 2012), cumulative, communicative, argumentative, ecological validation (Sarantakos as cited in Bapir, 2012) to name but a few. This has led to qualitative scholars coming up with some guidelines of validity and reliability in qualitative research which are different from their quantitative counterparts.

In the current study, issues of trustworthiness, credibility, and transferability were considered by conducting a pilot study, conducted in three organisations in the Gauteng province of South Africa. The main aim of the pilot study was to test the research instruments and to familiarise the researcher with the software development environment. After the pilot study, the instruments were adjusted and corrected where necessary and were prepared and adopted for the final study.

4.7. Data collection procedure

Data were collected from March 2015 to December 2016. At first, a conditional ethical clearance was issued and it was used to start the data collection phase. Data were collected through face-to-face interaction with the managers and the developers. Appointments were arranged with respondents and the researcher visited their premises.

Before the interviews started, the researcher gave background information about the research, its aims and objectives, then gave the respondents the informed consent form. After reading the informed consent form, the respondents were asked to sign it and the interview started. Respondents were also asked to give consent for the interviews to be recorded. In most instances, the interview questions were sent to the respondents before the actual interview. The interviews were semi-structured, allowing respondents to give as much information as possible but also being guided by the structure of the questions.

4.7.1. Research instruments

One of the characteristics of case study research is that data must be collected from a variety of sources. Yin (2014, p. 105) lists six sources of data in case study research. They are documentation, archival records, interviews, direct observation, participant-observation, and physical artifacts. In this study, data were collected from two sources: IT/project managers

and software developers. Different research instruments were used to collect the data. The instruments are discussed below.

4.7.1.1. Interviews

Nieuwenhuise (2007, p. 87) defines an interview as a two-way conversation in which the interviewer asks the interviewee questions about the issue under investigation. Interviews are aimed at collecting ideas, views, opinions, perceptions, behaviours and beliefs of the respondents. According to Nieuwenhuise, interviews can be open-ended, semi-structured, and structured. Creswell (2009, p. 179) lists a number of ways in which interviews can be conducted. According to Creswell, they can be conducted face-to-face, by telephone, by email and in groups. For the purposes of this study, semi-structured interviews were conducted face-to-face. In semi-structured interviews, questions are developed before the actual interview, but there is room to ask further questions if the need arises. That is, the interviewer has prepared questions in advance but can also probe further for answers, asking follow-up questions. Semi-structured interviews were conducted with IT/project managers and software developers. IT/project managers were interviewed to determine how the whole organisation managed its knowledge assets and developers were asked how they manage their knowledge at individual level.

Two interview schedules (See Appendix A for the IT/project managers and Appendix B for the developers) were developed for the interviews. The interview schedule for project managers had three sections. Section one had questions about software development failures and their causes. Section two contained knowledge management constructed from the knowledge management conceptual and theoretical framework developed for the study. Section three contained questions about post-mortem reviews, knowledge management benefits and challenges. The interview schedule for developers had two sections. Section one had questions about knowledge management practices and Section two had questions about

knowledge management benefits. Knowledge management practices questions were constructed from the conceptual and theoretical framework developed in the study.

A tape recorder was used to record the conversations. The interviews with project managers lasted for about 60-90 minutes each and interviews with developers lasted about 30 minutes each.

4.8. Research horizons and research time frame

Two research time horizons are common in research: cross-sectional and longitudinal time horizons (Hussey & Hussey, 1997). Cross-sectional studies collect data just once over a period of time. Longitudinal studies collect data more than once, over time. The study applied the cross-sectional research horizon. Data were therefore collected once.

4.9. Data analysis

Content analysis was used as a data analysis method. According to Krippendorff (2004, p.3) content analysis "is a systematic reading of texts, images and symbolic matter through classification, tabulation and evaluation in order to ascertain its meaning and probable effect". It is a method of categorising written or oral materials into similar meanings (Moretti *et al*., as cited in Cho & Lee, 2014, p. 3). The same sentiments are shared by Elo and Kyngäs (2008, p.109) who state that the goal of content analysis is to group large quantities of data into smaller categories. They further state that content analysis can be quantitative or qualitative, deductive or inductive (p, 107). This study adopted inductive qualitative content analysis, and adopted the steps proposed by Elo and Kyngäs (2008, p. 110). Content analysis was adopted because of the nature (text)of the data collected and because the data were supposed to be categorised into themes. Content analysis enables that.

There are three major steps involved in inductive content analysis, namely preparation, organising and reporting. Before the data were analysed, they were transcribed from voice to written text. After the transcription came the first step of preparation as explained by Elo and Kyngäs (2008). During this stage, the transcribed data were arranged according to the organisations they were obtained from. For each organisation, there were two data sets; one for managers and the other for developers. Upon the completion of the transcription, the researcher had to go through each transcript to make sure that they were transcribed well. Where errors were identified, the researcher went back to listen to the recording and made the necessary corrections. After the verification process, the data were then organised.

The data organisation process involved open coding, creating categories, and abstraction. The open coding process entailed taking notes, marking and highlighting important sections of the transcripts. For example, in this study, text sections that were about software failures and knowledge management practices were highlighted for further interpretation. From the open coding exercise, categories started to emerge and were written down. Further analysis reduced the categories from broad to more specific narrower categories (abstraction).

The abstraction phase produced two main categories; software development failures and knowledge management practices. The software development category has two subcategories; types of failure and causes of failure. The knowledge management category has four subcategories, knowledge gaps, knowledge management practices, knowledge management benefits and knowledge management challenges. These categories are discussed in detail in the results chapter.

4.10. Logic of reasoning: Inductive reasoning

Blaike (2010, p. 81) presents four research approaches (logic of reasoning) that are used in social research. They are induction, abduction, deduction and retroductive research. Induction aims to establish descriptions of characteristics and patterns. It aims to answer ‘what’

questions (Blaikie, 2010, p. 84). Collis and Hussey (2003) state that such studies develop theory from observation of empirical reality. They define deduction as a "strategy to find an explanation for an association between two concepts by proposing a theory, the relevance which is tested" (p. 85). Deduction seeks to answer 'why' questions. It is defined by Saunders, Lewis and Thornhill (2009, p. 124) as involving the development of a theory which is then subjected to rigorous testing. Saunders *et al.* (2009) further state that deduction involves explaining causal relationships between variables, developing and testing hypothesis, and allows the generalisation of results. Retroductive research aims to "discover underlying mechanisms that explain observed regularities in particular contexts. It is a process of building hypothetical models of structures and mechanisms that are believed to generate empirical phenomena" (Blaikie, 2010, p. 87). It involves working back from data to a possible explanation. Abduction involves developing theories derived from social actors, meaning in the context of everyday activities. It begins with the description of such activities and meanings and deriving categories and concepts from them that can form the basis of an understanding of the problem at hand (Blaikie, 2010, p. 89). The study adopted induction. This is because theory was built after empirical data were collected, analysed and interpreted. The next section discusses the research methodology of the study.

4.11. Access, privacy, confidentiality and ethics

Every researcher is expected to consider research ethics as he/she conducts research. In any research involving people, respondents are protected from any form of harm. The University of KwaZulu-Natal (UKZN) has a code of research ethics that it expects every researcher to adhere to. This research was conducted in conformity to the UKZN code of ethics. Permission was sought from participating organisations to conduct the study and permission was granted before the research took place (See Appendix F for letters requesting access and permission letters). Respondents were notified that participation was voluntary and that they

could withdraw at any time of the study without any form of sanctions. The informed consent form contained information about the rights of the respondents. Before the interviews were conducted, the researcher read or gave the interviewee the consent forms for consideration. When the interviewee agreed with the conditions, they were made to sign the consent form. Privacy and anonymity of organisations and people was observed. Information obtained from company documents was kept safe and used for the purposes of this study only (See Appendix G for informed consent forms).

4.12. Summary

This chapter discussed the research design. It discussed the philosophical perspectives underpinning the study, the logic of reasoning adopted, the research methodology and research method, the population and sampling procedures, research instruments, research time horizons and issues of validity and reliability. Drawing from the software development literature, it is evident that qualitative case studies are conducted most. This study followed that trend. Table 9 provides a summary of the research methodology and design.

Table 9: Summary of research design

Scientific method	**Choice description**
Philosophy	• Ontology - relativism • Epistemology – interpretive
Research methodology	Qualitative
Research method	Case study
Validity and reliability	Pilot study
Logic of reasoning	Inductive
Research time horizons	Cross sectional
Population	Software development SMMEs in the province of KZN
Sampling procedure	• Sampling method - Non-probability • Sampling technique – purposive and snowball
Data collection procedure	• Interview schedule
Data analysis	Content analysis

CHAPTER FIVE
DATA ANALYSIS AND PRESENTATION OF FINDINGS

5.1. Introduction

This section presents results obtained from 12 IT/software project managers representing the 12 organisations who participated in the study. Eight software developers from the 12 organisations were also interviewed and their responses are presented in this chapter. This chapter has two sections. The first presents data from IT/project managers, the second presents data from developers. Data from managers is divided into two sorts: software development failures and knowledge management practices, benefits and challenges. To ensure confidentiality, pseudonyms are used for all respondents and organisations. IT managers are called Manager 1-12 respectively and developers are called Developer 1-8 respectively.

5.2. Software development failure in organisations

The first research question was to investigate whether their organisations encounter software project failures. As indicated in earlier chapters (in more detail in Chapter 3), the issue of software development failure is a serious problem in software development organisations. It is one of the reasons software development organisations adopt knowledge management practices. This question was aimed at finding out if, currently, software organisations are facing failures, which would then justify their adoption of knowledge management.

When asked if their organisations have encountered project failures, the managers responded as follows:

Manager 1 observed, "*just one particular case, our current developer took over from another developer who left, who developed a backup which was not really successful. He had to redevelop it.*"

Manager 2 asserted, "*yes we have but not many. We have had a lot of delays, and cost over runs. I have also pulled out from three clients myself*."

Manager 3 observed, *"no because we are not developing new [software] products, but we are enhancing an already developed product."*

Manager 4 noted, "*no, I can't say that, I am new in the organisation*."

Manager 5 replied, "*sure yes! We have had cases in the past. Due to lack of resources from our part, we were unable to complete the project in the time that was required and not according to the client's specifications*."

Manager 6 concurred, "*yes! We created software three years ago with another guy who later left. When he left, we had no documentation. I had to basically start from scratch again.*"

Manager 7 acknowledged, "*you always get failures, but there are two sides. One is incorrect knowledge and when you give them[clients] the software, they suddenly say it's a failure because the goalpost has moved.*"

Manager 8 responded, "*yes, I have abandoned projects myself. A client of mine invested a huge amount of money in the project and infrastructure. The client did not do an information study on his business and eventually we agreed to kill the project even though it could have made a huge amount of money. He could not finance it.*"

Manager 9 said, "*not really, some of them did not go further than expected because the space was just too vague and the applications were not polished enough and energy and resources were not enough.*"

Manager 10 pointed out, "*yes, it happened once. Somebody advised us to develop company software. We did, and it was perfect. You know what, it never sold and we think of scrapping it.*"

Manager 11 replied, "*we have not had any project that did not succeed, but we have had a project that we incurred a lot of costs.*"

Manager 12 stated, "*absolutely! There were quite a number of projects where you find that were are developing and you find that the client is not happy about and then we have to stop. There was also an instance whereby we maintained a project for a couple of years and then the client would disappear because they are no longer impressed with the product."*

The results indicate that the majority of the organisations have experienced software development failures in their lifetime. Only two organisations indicated that they had not had failures. Manager 3 indicated that this is because his organisation does not develop software from scratch but customises already existing software. Manager 4 indicated that he is new in the organisation and could not confirm nor deny that failures have happened before. The results indicate that software development organisations do experience software development failures.

5.2.1. Types of software development failures

Further analysis of the results from the managers were conducted to determine the types of failures that the organisations encounter. This was done with reference to Chapter two (types of software development failures). As stated previously, the study adopted the two

classification types of failures by the Standish Group CHAOS Manifesto (2013). Table 10 shows the types of failure or failures found in each organisation.

Table 10: Types of software development failures

Organisation	**Type of failure(s)**	**Cause of failure (according to conceptual framework)**	**Managers' comments**
One	Abandoned	Technical issues (software and hardware compatibility issues)	Developed a backup which was not successful. Had to be developed again. Had compatibility issues.
Two	Challenged and abandoned	Organisational and external factors (management, organisational culture, lack of skills and market and client driven issues)	Had delays, cost overruns and abandoned clients.
Three	No failures	N/A	Organisation deals with product customisation only.
Four	No failures	N/A	Not sure because they are new in the organisation.
Five	Challenged and abandoned	Project management issues (failure to manage projects)	Project was delayed and with less specifications required by client.
Six	Challenged and abandoned	Project management issues (failure to manage projects)	Projects are sometimes late and one was abandoned and had to be redeveloped.
Seven	Abandoned	External issues market, competitor and clients driven issues)	Product rejected by clients.
Eight	Abandoned	External issues (market, competitor and clients driven issues)	Projects abandoned
Nine	Abandoned	Organisational issues (management, organisational culture issues, lack	Projects abandoned

		of skills)	
Ten	Abandoned	External issues (market, competitor and clients driven issues)	Lack of interest from clients.
Eleven	Challenged	Project management issues (failure to manage projects)	Incurred a lot of costs
Twelve	Challenged	External issues (market, competitor and clients driven issues)	Product abandoned by clients.

The results indicate that the projects either failed completely and were abandoned or were challenged. It is worth noting that some managers indicated that they did not have failed projects but their comments indicate that they had either failed or challenged projects. This applies to organisations nine and eleven. The results indicate that the majority of the organisations have had either a failed/abandoned project or a challenged project or both types of failure.

5.2.2. Causes of software development failures

Apart from identifying the failures and the types, managers were asked to give their opinions on what caused the failures. Ten causes were found. They are bureaucracy in IT departments, compatibility issues, complacency of developers, involvement of wrong people at the planning stages of the project, lack of detailed documentation, lack of resources, lack of user commitment, miscommunication, unrealistic customer expectations, and work overload. The causes are discussed in detail below.

5.2.2.1. Bureaucracy in IT departments

The issue of bureaucracy in IT departments was raised by manager 2, who noted, "*the problem is bureaucracy on the part of IT departments. I do not understand why*." He further said, "*I do not understand the security sensation about that*." His statements make one assume that bureaucracy caused problems. Such problems are delays either in starting or completing the project because of resources, organisational politics and other reasons.

5.2.2.2. Compatibility issues

Manager 1 indicated that in their case it was compatibility issues that prevented the software program from working. He said, "*I don't know exactly [the cause of the failure] but I think it had compatibility issues with Linux.*" No other manager mentioned such a cause, but it is clear that it does cause failures.

5.2.2.3. Complacency of developers

According to the respondents, complacency is common among senior developers. This is because experience and seniority make them rush because they assume that the projects they developed previously are similar to current projects. Manager 8 stated that, "*Most programmers come with experience. The more experience they have, they tend to rush. They think that the projects they have developed prior are similar to the current project.*" Manager 4 concurred and said, "*The other reason is team members not doing their work well and the work is concentrated on one or few guys.*" This statement is interpreted as complacency among team members.

5.2.2.4. Involvement of wrong people in the planning stage of the project

Manager 8 revealed that in some cases, customers who have no development knowledge get involved in the planning stages of projects. "*This is a task for developers,*" he said. He further gave an example in which a business person with no development knowledge was personally involved prematurely in the planning stage of a project and put the developers under pressure. "*The developers started developing even without properly getting the specifications of the project. The project was bound to fail even before it was started,*" he lamented.

5.2.2.5. Lack of detailed documentation

Manager 2 indicated that a lack of documentation during the specification stages of the projects causes problems because they tend to forget or misinterpret what the customer actually wants and develop something that is completely different from the actual specifications. In his words, "*sometimes it is the lack of detailed documentation during the specification stage [that causes failure]".* Manager 6 gave an example in which a developer who was involved in a project left the organisation without the documentation of the project. "*When he left, we had no documentation at all. I had to take it over [the project] and I had to basically start from scratch again,*" he said.

5.2.2.6. Lack of resources

The study also found that a lack of financial and human resources is another cause of failure. When asked if they had any failure and, if so, what the causes were, Manager 5 replied, '*due to lack of resources from our part, we were unable to complete the project in time and not according to the client's specifications*." His words were echoed by manager 6 who said, "*The lack of skills was the main cause [of failure] because the [skilled] guy left.*" Manager 6 also added to the issue of financial resources and said, "*there were certain things that were supposed to be in the website, but the budget for the website did not amount to those.*"

5.2.2.7. Lack of user commitment/non-use of completed software

It was also established that after the system has been developed, users sometimes refuse or are reluctant to use the system. This was confirmed by Manager 2 who explained, "*the other cause [of failure] is lack of user commitment.*" The same sentiments were shared by Manager 5 and Manager 10 who indicated that sometimes the client does not deploy the systems which then amount to none use. "*In some instances you find that we have done the development but it has not been taken over by the client on their side. The customer has not implemented it,*" lamented Manager 5. Manager 10 added, "*we developed a program but it just did not sell.*" The last statement explains a system that has been developed but was not accepted by the users or the market.

5.2.2.8. Miscommunication/misinterpretation of requirements

Managers indicated that sometimes there is lack of communication or misinterpretation of requirements at the beginning or during the course of the development process. This causes failure because the systems might have unwanted functionalities or might not have all the required functionalities. In their own words they said, "*the major causes of failure during a project is lack or bad or poor communication during the project conceptualisation*" [Manager 2]. Manager 4 added, "*The reason it failed was that we were trying to answer questions that we did not understand.*"

"These guys started developing even without properly getting the specifications of the project, and to make this worse, they are Spanish speaking and there was a huge amount of miscommunication", lamented Manager 8. The statements point out to one thing, miscommunication as a cause of failure.

Manager 12 said, "*there would be lack of understanding from the client on what the full requirements would be and it could be rush! Rush! Rush!"*

5.2.2.9. Unrealistic customer expectations

Three respondents, Managers 2, 7 and 8 stated that this is a serious issue. According to them, clients and development organisations sometimes have unrealistic expectations of the product. They expect the software product to have functionalities that are not possible to have. Manager 8 explained, "*sometimes the client does not differentiate between real needs and perceived needs."*

Manager 7 added, "*when you give the customer the software product, they suddenly say it's a failure because there are more things expected because the goalpost has moved*".

Manager 2 echoed the same sentiments and said, "*on our part, I think I led the client to believe that we can address all their needs without checking what the unstated needs were*." This manager admits that it was his organisation that did not clearly define the user expectations.

The same concerns were raised by Manager 12 who lamented, "*the biggest thing [problem] came from the client, trying to pull all the information but as you develop they change and say this is not what I want*."

5.2.2.10. Work overload

Manager 8 indicated that sometimes developers feel overwhelmed by their workload. He said, "*software organisations operate like hospitals. They accept as many projects as possible because they are afraid to lose the business. They take the deposit and fail to deliver the product on time.*"

This was echoed by Manager 12 who agreed "*when I first came here, there were too many projects being handled at the same time and we had a small team of developers. I think we took more than we could handle and that caused stress to the developers. When that happens,*

too many problems arise and that leads to poor quality code and maintenance and features not being built in properly."

The results point out ten major causes of software development failures. Surprisingly, only two are knowledge management related; lack of skills and lack of documented knowledge. The rest are caused by factors unrelated to knowledge management.

5.3. Knowledge management practices in software development organisations

The main purpose of the study was to investigate knowledge management practices in the organisations. This section presents results of the knowledge management practices of the organisations. Knowledge management practices questions were derived from the conceptual and theoretical framework (chapter two). This section is structured according to the framework, but also presents results about knowledge gaps encountered by organisations, the benefits of knowledge management and the knowledge management challenges organisations face. Results of knowledge gaps are presented first.

5.3.1. Knowledge gaps encountered by software organisations

Managers were asked to state if they encounter knowledge gaps in their daily tasks in their organisations. This question was asked to determine if organisations experienced knowledge needs which would in turn lead to the adoption of knowledge management.

When asked if software development organisations encounter knowledge gaps, the managers responded as follows:

Manager 1 noted, "*I don't think so in particular, maybe if we have new employees they might struggle to get new information and get to speed with the rest of the guys who have been working with us for a long time.*"

Manager 2 observes, "*we are a Microsoft partner and we get most of our information from them and social groups within the Microsoft world. I think we make it our problem to find it. So we tend to pull the information [from Microsoft].*"

Manager 3 agreed, *"yes, it does happen a lot. There is business knowledge and there is also development knowledge that we usually seek."*

Manager 4 also concurred. He said, *"all the time. For example, you are trying to write a piece of code in a class but you don't know how the logic is supposed to work. You have the structure but the logic is not appealing to you."*

Manager 5 shared the same sentiments when he said, *"yes we do. With software development there are new technologies and in our case there is hardware we need to support. For example, a new printer and you have to find information about it."*

Manager 6 explained, "*yes, as a person and organisation you don't always know everything. Definitely there is some kind of knowledge gap in certain things*."

"The knowledge management gap comes in the interpretation of information provided by one person to the next, so it's the old game of Chinese whispers. What I tell you, you tell your associate and then he tells his associate. By the time the information reaches you, it has changed", narrated Manager 7.

Manager 8 said, "*all the time. I spend a lot of time myself doing research. My essential philosophy is that unless you have done the research, things are going to fall*."

Manager 9 agreed, *"yes we do. There are areas that I had identified. When we wanted to move to a GIS [Geographic Information System] system, I had never worked with the GIS system before. I did not know the language etc."*

Manager 10 explained, "*yes we do. You must have knowledge about your subject and knowledge about your user. You must also know your market."*

Manager 11 asserted, "*yes, definitely. We often work on new projects that require new information. We learn each time from them. We do research on that project.*"

Manager 12 noted, *"yes, when we are faced with new challenges in projects. We usually seek knowledge on new software development techniques, coding, etc."*

The results from the respondents indicate that companies encounter knowledge gaps or knowledge needs. They indicated that it is not possible to know everything. The responses present different scenarios in knowledge is required, but what is clear is that the knowledge is needed for the daily activities of the organisations. In summary, organisations encountered knowledge gaps when they start new projects, during software development, when new technologies are introduced, knowledge gaps about the industry they are developing for and their clients. The results indicate that software development organisations encounter knowledge gaps.

5.3.2. Knowledge acquisition

After it was determined that organisation encounter knowledge gaps, the next question that was put to managers was, where do they get the knowledge to fill their knowledge gaps from? That is, what knowledge sources do they use to acquire knowledge. The results indicate that knowledge is acquired from a number of internal and external sources.

When asked where they acquire knowledge from, the managers responded as follows:

Manager 1 indicted that it is mostly new employees who require knowledge and that they acquire it from seniors. He said, *"obviously, they have to ask senior people in the*

organisation for assistance. It could be other colleagues that we work with, or it could be suppliers."

Manager 2 explained, "*we are a Microsoft partner and we get most of our information from them and social groups within the Microsoft world. So we tend to pull the information [from Microsoft]."* He continued and said, "*you simply go to the user groups or Google user groups or straight to Microsoft development network and look at things there and then you find information.*" Apart from Microsoft, he indicated that there are other sources they use. "*There is also a YouTube website, and the other one is just to put the question on Google*", he added. He continued to explain that they also get knowledge from their clients and industry peers. He said, "*there are packages that we have written and sold, and constantly we develop them based on information we get from our clients. Client base is a very rich source of information for us. We also have an industry network. Basically, it's a user group in which we exchange information. We give them information and they give us back, for example, if there is a change in legislation.*"

Manager 3 mentioned that their main source of knowledge is their data warehouse. The data warehouse contains technical and business information. For example, development manuals, client information, and general business information. He stated that, "*we have a data warehouse with books, forums, articles etc., and that is what we use mostly.*"

Manager 4 explained, "*knowledge is acquired through brainstorming, and workshops, but I prefer self-study and self-improvement.*" In addition to that he said, "*some of our stuff members were taken to the USA where all the technologies of the future are presented. That's how it works*." He continued to explain how they use videos for knowledge acquisition purposes. "*you can find videos online and its usually helpful. Nowadays our wonderful smart*

phones have applications that you can study from and share stuff with your friends and basically I think the information is in the World Wide Web", he added.

Manager 5 was of the view that, "*The Internet – we Google the information. There is also a network that we use. My boss is part of an entrepreneur network. He goes to those meetings and learn from them.*"

Manager 6 noted, "*we get other developers to actually help us. We use things like Skype and social networks to communicate. If we don't know something, the first thing that we do is ask the guys that are here if they have come across the same problem before.*"

Manager 7 observed, "*you have to extract the knowledge from the customer because that is where it is*." He further explained that, "*the programmers usually help each other when they have an information gap relating to their specific work areas. For example, a problem with a programming tool; that's when they ask each other questions*."

Manager 8 said, "*Google is my first stop. I can't imagine life without the Internet*. *I also subscribe to a whole lot of leading e-magazines in America and all over written by specialist who are doing writing of such articles about where the industry is at the moment, and who are the customers. That's where I get most of my information from."*

Manager 9 explained, "*the best way to acquire knowledge is to teach yourself, but sometimes we have to outsource the tasks and ask someone to do it for us. Sometimes we ask each other questions and explain how to do it and let you struggle with the problem otherwise you never learn. We also have training sessions here. We bring the staff together and share content. We also use the Internet a lot especially for standards and specifications."*

Manager 10 noted, "*everything is on the Internet. You go to the Internet and you find everything there. All the latest tax laws and amendments are all there."* He continued to say

"Nick [the other developer] has a Visual Basic user group in Johannesburg where they meet and share knowledge."

Manager 11 pointed out that, "*Google!!we do a lot of online research, searching a lot of websites. There are a couple of websites that we use on a regular basis and amongst ourselves we discuss and ask questions.*" "*At times we work with other IT departments and ask them questions. For example, we don't work with oracle database…so we ask others to help us*", he elaborated.

Manager 12 elaborated, "*we acquire knowledge internally and externally. Internally from other staff members and externally from the Internet and other fellow developers in our field [industry]."*

The results indicate that software companies acquire knowledge from a number of internal and external knowledge sources. Internally, they acquire it from each other and externally, they use the Internet, clients and industry networks. The Internet is the main source of knowledge for the companies. Almost all managers stated that they use it quite a lot. The conclusion therefore is that software development companies acquire knowledge from internal and external sources and that the Internet is the main source of knowledge.

5.3.3. Knowledge creation

Apart from acquiring the knowledge from different sources, managers we asked to indicate how they used the knowledge sources they stated earlier to create their own knowledge to be used in their tasks. They said,

"It is usually staff meetings and brainstorming sessions." [Manager 1].

"We work directly with Microsoft and that's where we get the information from, but otherwise we rely upon Internet based tutorials." [Manager 2].

Manager 2 was further asked if his company provides training, attends workshops and conferences. He responded, "*we do provide one-on-one training to get close to new people."*

Manager 3 stated that they use the information stored in their data warehouse. They look for relevant information, refine it and internalise it and then use it in the tasks.

"If I am writing an application and then I get stuck, I am going to go to the Internet and research something and I am going to find more information. I will attempt to put into the program and if it works, obviously I am going to show it off. I would say as individuals we also study on our own at home just to learn something new." [Manager 4].

Manager 5 said, "*we attend conferences and things like that.*" He was asked if the organisation provides or sends employees for training to create new knowledge, he said, "*Not really."*

"Most of the time we download online videos and learn from them to keep up-to-date with software trends", said Manager 6.

Asked if they attend training, seminars and conferences, he replied, "*at the moment we have not been but if we need help we ask someone to teach us."*

Manager 7 explained, "*we attend training courses that we perceive will benefit one of our programs. We also do internal training. We also encourage the developers to spend a few hours on the Internet doing online research. We also try to encourage them to look where the IT industry is going and try to upskill themselves in the different skills sets.*"

Manager 8 pointed out, "*we spend a lot of time self-training, upgrading our skills. Developers are trained in development and business skills.*"

Manager 9 stated that, "*we use the Internet information a lot*."

When asked if they conduct formal training and attend seminars and conferences, Manager 9 replied, "*we don't often go for training, we normally train ourselves because we have this engineering background. Courses, we do occasionally. It has been ten years since I did the last course. Training is more of an option.*" When asked if they attend seminars and conferences he said, "*yes, but the purpose is not to acquire knowledge but to acquire the market*."

Manager 10 observed, "*I use the Internet which has tones of information and its tells you exactly what to do. I also go to my clients who are knowledgeable and advise what to do*."

When asked if they conduct or attend trainings, seminars and conferences to gain knowledge he responded, "*Nick did a couple in the beginning, but everything that we need to do up-to-now he has handled quite comfortably. He always finds ways of doing things around*."

Manager 11 said, "*we have formal training sessions. Sometimes we assign a person to research a topic then we will sit for a few hours and get feedback.*"

Manager 12 asserted, "*we have brainstorming sessions whenever there is a new project to take on. We bring in senior people from the various divisions and everybody gets to input for the suitable solution and based on that we architect the blueprint of the project."*

When asked if they conduct or attend formal training, seminars or conferences he replied, "*we have in the past. My lead developer has been to [Las] Vegas to a conference and has done all the recommended training and has passed over that knowledge to the rest of the team."*

The results indicate that, organisations create knowledge using a number of methods. The most common one is reading and internalising Internet information. Other methods are self-study, consulting clients and brainstorming sessions. It is surprising though to note that

organisations rarely send employees to workshops and formal training sessions. This is the most common method of knowledge creation and it is not fully utilised by organisations. There is no evidence of research as well.

5.3.4. Knowledge storage

The next question that managers were asked was whether they store their knowledge resources, and if they do, how do they store them. The results indicate that they store them mostly electronically in organisational repositories or servers. When asked if they store their knowledge and how they store it, managers responded as follows:

"*We store our information in the server",* said Manager 1.

Manager 2 replied, "*we store it in the server and we use Dropbox to move things around."*

Manager 3 stated that, "*we have HAPPY FOX, our institutional database. We also have a data warehouse where we have books, manuals, and discussions with people*."

"*We have a portal in the server and FTPs [File Transfer Protocol] where we store our information*", said Manager 4.

"We *have a software called conference. It is linked to our compile tracking software; a document storage which is hosted in the server. We have membership [mentions name of organisation] and they have things like books and videos which are available online*", explained Manager 5.

Manager 6 stated that, "*in the server and emails. We have got an external hard drive where we back up everything, it can be accessed through the network."*

Manager 7 said, "*we have several different knowledge repositories, using disparate systems. Some we use SharePoint, some on peoples' spreadsheets on laptops and some on email and*

workflow systems; all the normal things that one controls." He continued to explain that there is no central repository to store knowledge. "*What we should is to have one central location and use SharePoint where everyone is depositing data into the repository and getting data out of the repository as required. That is what we should be doing but we do not,*" he added.

"*We store everything in a secure server. We store it electronically so that it does not get lost and it is easier to retrieve it",* explained Manager 8.

Manager 9 explained that they have a database in the server where their information/knowledge is kept.

Manager 10 stated that, "we *have a website and web server that we rent and put all the video files on the cloud. We rent web space from a company in Salt Lake City*."

Manager 11 explained, "*we keep [our knowledge] in my Outlook folder in an archive of all help articles that the guys have sent out. We don't have a central repository and I suppose if we had a central repository, that would be me.*"

Manager 12 said, "*we use a number of internal tools. We have got an internal development Wiki that we keep. We also check [deposit] our source code into GET [an internal storage repository]. We store electronic tutorials, PowerPoint presentations, YouTube videos, images and our source code in GET too."*

The results indicate that organisations store their knowledge in a number of storage devices. The majority use computer technology. Most store their knowledge in servers although others indicated that is it scattered all over the organisation in people's computers and one indicated that it is stored in the cloud outside the organisation. Within those computer systems, it is stored in different folders and file systems. The results show that organisations store their knowledge resources internally and externally, using different storage devices.

5.3.5. Knowledge organisation

After storing their knowledge resources, managers were asked how they organise their knowledge resources in the devices they use. This question was asked to determine how easy it is to retrieve because the assumption is that the easier it is to retrieve, the quicker and easier it is used. In their own words the managers said,

"We organise just a couple of documents in the server in folders." [Manager 1].

"We organise the information in the server. We have folders with associated products and in these folders they will be associated with technical references etc." [Manager 2].

"*The information is organised in the data warehouse according to its categories*." [Manager 3].

"*Our information is organised in our portal. We have different folders there.*" [Manager 4].

"*Our database has different information, for example, coding standards and guidelines, and different projects*". [Manager 5].

"*Everything is organised in terms of folders. Relevant documents pertaining to something, for example, if something has to do with our training setup, that document goes to a specific folder, if it is something to do with programming it will go to the programming folder."* [Manager 6].

Manager seven: [information not organised because it is scattered all over the organisation]

Manager eight: [response was not addressing the question posed]

"*We have specific folders, especially software development folders and examples of source code, mathematical models, excel models. It is documented that way*." [Manager 9].

"*We hardly organise our information here because we get everything from the Internet*." [Manager 10].

"*Not formally at all. What we would do is a checklist and procedure we follow*." [Manager 11].

"Our knowledge base is broken down into sales, pre-sales, project management and development. In each of those we have information stored specific to that issue." [Manager 12].

The results indicate that organisations organise their knowledge resources in their servers, organisational databases, repositories and portals using folders.

5.3.6. Knowledge sharing

The next question that was posed to managers was how they share or disseminate the knowledge within and outside the organisation. This question was asked based on the premise that knowledge that is not shared has little value within the organisation.

Manager 1 said that, "*it is usually staff meetings, otherwise emails. Sometimes it is brainstorming sessions and other discussions which are mainly handled by the development and technical teams. This is usually done in the morning."*

Manager 2 observed, *"I get close to new people and I expect everybody to help them."* This statement is interpreted to say that the manager shares his knowledge mostly with junior employees.

Manager 3 said, "*we share and disseminate information amongst ourselves and it is mostly face-to-face.*"

Manager 4 stated, "*we come together and share ideas. For example, someone would say, hey did you know about this, which is good you know because it keeps the team alive.*"

Manager 5 explained, "*within the organisation it is rarely done. We rely mostly on external sources.*"

Manager 6 asserted, "*we usually share it using flash discs or via the network.*"

Manager 7 said, "*the developers help each other when they got a knowledge gap relating to their specific work area*." He further explained that his company also transfers knowledge to young developers. "*we have what we call developer factory. We take young students that we believe are having the capacity to learn programming and train them to our best ability*", he added.

Manager 8 observed, "*I would have information that is publicly available and send [email] it to the developers and we discuss later face-to-face. Our clients get it from the server.*"

Manager 9 noted, "*if we have a problem we could say, let me help you solve it. We sit together and talk it through. When someone has a training session, we would bring all the staff together and show them the new concept*."

Manager 10 said, "*He calls me all the time. We use the telephone and email"* he continued to narrate how they communicate to share knowledge and he said*, "I said Nick, this is what we want and here are the formulas. He sent it to me by email and I added into the software and sent it out as an update to my clients.*"

Manager 10 pointed out, "*we have a lot of meetings to design internal processes. We also have a lot of brainstorming on how we deal with situations*."

Manager 12 explained, "*all internal communication that we have between people is communicated through a platform called SLACK. Everything happens in SLACK in terms of communication. All the instruction, pieces of development and requirements are managed in SLACK"*. He also explained how knowledge is shared with customers. *He said, "customers have different access. We store their information but we sometimes get customer enquiries via email and WhatsApp. We take those communications and copy them into SLACK."*

The results indicate that organisations shared or disseminated their knowledge internally and externally using a number of methods. Externally, knowledge is usually shared with clients. They get it mostly from the organisational servers using network technologies. Internally, knowledge is shared mostly using face-to-face interaction and email. Face-to-face is usually during meetings, brainstorming sessions and during informal one-on-one training sessions. There is no evidence of publications (for example, industrial property, patents, trademarks; conferences/seminars, etc.)

5.3.7. Knowledge application

Knowledge that is not applied where it is needed has no value. Managers were asked if they use the knowledge resources at their disposal and were asked to indicate how they applied it. All managers indicated that knowledge was applied in their daily tasks. In this question, they were requested to narrate a case in which organisational or external knowledge was used in their tasks. They responded as follows:

Manager 1 narrated, "*One of the modules we have is a SMS [short messaging service] facility. It is an option that not all the customers have. Basically, a customer requests that we quote them and we install the application. The lady who used to do it left and we used the documentation in the server to install it."*

Manager 2 stated, "*It is post-mortems' information that helps us a lot. In post-mortems, we ask what we did wrong and right. That information helps us in the future.*"

Manager 3 narrated that in most cases they use the knowledge that is stored in their data warehouse for their development and business related purposes.

Manager 4 explained that, "*for the licensing part, our senior programmer read about AZURI a while ago. He later realised that you can put the license in AZURI. Some of the stuff that he learned was from the USA conference.*"

Manager 5 said, "*Yes! in the case that there are specific bugs that have been reported, we will generally report where it happened and how it happened, and generate an error message and store it. We will then take that and search it online, have suggestion and then use that in the bug fixing.*"

Manager 6 explained, "*in terms of programming [knowledge], we put comments after each programming code and we document the code. Our code is object-oriented and it can be used somewhere else. Basically if we need to use the object in other software programs, we could copy this code across because we can see the logic and just basically change a few things in that program. That is how it is documented and used.*"

Manager 7 pointed out that, "*every time we finish a project, we look at estimated costs and time taken to complete the project then we take that information and apply it to the next project.*"

Manager 8 said, "*we use the knowledge we have in the server for development and business purposes.*"

Manager 9 explained, "*let's say I develop a software model of how you simulate a power transformer, all the mathematics and all that. That will be stored in a certain location in the*

source code and any developer can log into that location and see what is there and replicate it."

Manager 10 narrated, "*I wanted to add a feature in our software program. I did not know anything about it and I had to do research on it. So I went to the Internet. I also went to see some of my clients who specialise in winding dead peoples' estates and asked them what they advise that we show there. It was the research that I did on the web and talking to experts on the subject that helped me*."

Manager 11 stated that, "*we do a lot of SQL [Structured Query Language] programming. Someone will come with a practice communicated. In fact, we feel fairly good about this when somebody has found something which is quite significant. We get the developers together and we discuss it and certainly going forward we will implement that as a practice*."

Manager 12 narrated, "*last week, we were working on a new proposal that went out to the customer to get an indication what the website would look like. With almost all the source code, repositories, configurations settings set in SLACK, as opposed to going out to my technical lead who does all deployments, me and Glen were working on that project, we took all those components ourselves, configured and deployed the system. It was an instance that saved four to five hours by us doing it instead of pulling developers, and technical resources."*

The results indicate that organisations applied the knowledge they have in their daily tasks. the cases they gave show different scenarios by which knowledge was used.

5.4. Benefits of knowledge management

After the knowledge management activities were determined, managers were asked if knowledge management has benefited the organisations in any way. The response was that indeed knowledge management has benefited organisations. In their own words, the managers said,

"It has benefited us a lot by saving time", [Manager 1].

Definitely! Absolutely! One encounters a set of particular conditions all the time that you struggle with. It is essential that the knowledge that you gain after solving that problem is stored somewhere so that other people have access to it in future. It is faster to go to the server and get that knowledge, [Manager 2].

"*Yes, the business would not have survived!",* [Manager 3].

"*When I moved here I found that there are more brains on the table. So basically being part of a team is more valuable than being on your own, but I think that if you share knowledge as a team you can progress to the next level which is happening a lot here*", [Manager 4]

"*In a small team it is difficult to know everything, but as the teams grows, knowledge definitely makes an impact. [The advantage is that] if I document it, I don't have to tell every new person and it saves my time not to teach every new person and it is my knowledge if I have written it down*", [Manager 5].

"*Yes definitely, it makes us more efficient in terms of the company. If there is a problem and none of us has faced that problem, we do research on how to solve that problem and we store that. If we run into a problem like two years down the line, in order for us not to waste time and trying to solve that problem again, we look for it through our content problems and find*

solutions to sort that again. We have become much more efficient and more productive as a company", [Manager 6].

"*Absolutely! Because now you have things like library of code that you can use in the server. So you are reducing reinventing the wheel and you are learning from using new techniques and you are learning from mistakes of using different tools*", [Manager 7].

"*Very much!! I adopted knowledge management when I was still a business manager; way before I started development. That is how this organisation has benefited from KM*", [Manager 8].

"*Absolutely!! Without this knowledge our business cannot survive. It is crucial and key to what we do. We are the only company doing this, so the IP [intellectual property] is key*," [Manager 9].

"*Yes!! I could not work without it [the knowledge]. The Internet is the most important one*", [Manager 10].

"*Definitely, without a doubt. A lot of the stuff we discovered through research has made us more efficient as a company and more accurate. There is a significant impact on the performance of the systems and it has been a number of those cases. There are certain coding techniques in which we coded in a certain way. Some guys discovered quicker ways of coding. We have collected the knowledge overtime and when new people come in that is useful for me because I usually go through all the stuff and use it as a basis for training*", [Manager 11].

"There are two other great benefits; we have retained our IP [intellectual property] in an environment that it is easily accessible. If anything happens to our staff, we have all the information in one place. I think that's the biggest benefit. From a timing perspective, as I

said earlier, we probably saved five hours of time and other resources which we were able to do without. We were able to pull information from the repository", [Manager 12].

The results show that there are a number of benefits that organisations reap from implementing knowledge management. Two major benefits are saving time and retaining their intellectual property. Other benefits that the organisations stated are that knowledge management has made them effective and efficient and that it has made it possible for them to reuse their knowledge and prevent the reinvention of the wheel.

5.5. Post-mortem reviews

Post-mortem reviews are regarded as a knowledge management tool in software development organisations. They are the best way to capture lessons learned and best practices thus facilitating organisational learning (Birk, Dingsøyr, & Stålhane, 2002; Dingsøyr, 2005). Managers were asked if they conduct such reviews in their organisations for the same reasons given in the literature. The responses indicate that not all organisations conduct them, in fact most do not. When asked if they conduct post-mortem reviews, managers responded as follows:

Manager 1 said, "*what usually happens is that when we are developing, someone in support test it [the developed system] because most of the post-mortem testing is done straight after development."*

Manager 2 stated that, "*we do, but I keep on losing money on this. It takes about three times before the results of a post-mortem are realised. Post-mortems definitely help. If we did not do them, we would definitely do the same mistakes continuously and because we have them, we do them far less regularly."*

Manager 3 noted that, "*It's done on a daily basis as opposed to a project. If we are adding a module, we would not sit down at the end and do a post-mortem, that is done in the meetings*

that I have mentioned. The idea is to ask, what did you do yesterday, what problems did you come across, what solutions did you find and what are you going to be doing today? And people then come in and say this is what I came across back then and this is how I solved it. They would go to issue tracker to throw in their knowledge there to tackle that problem."

Manager 4 revealed that, "*at the moment no, but we have our weekly scrums, where we ask the progress; that is, where are you? what did you do?*"

Manager 5 said, "*we do, but usually when there is something that really went wrong; then we do that. I don't think we do enough. When things go okay, we don't even go back to look at the things that we did right. So, it is usually only reactive in the case of things going wrong.*"

Manager 6 explained, "*we don't sit and review; we just have general meetings. That is where we finalise the project. It is basically after all testing that we obviously put all the problems that we found. After our testing phase, we have a review but we don't usually sit down and say what went wrong and how could have sorted it out.*"

Manager 7: [question was mistakenly omitted]

Manager 8 narrated, "*yes we do! Ours is not only about the code but the whole project management. We look at where issues developed in the project, how we could have avoided them and what could have been a better way of finding the solution. That has helped us a lot by not repeating the mistakes.*"

Manager 9 said, "*not formally, but I would sit with my developers and discuss the challenges we have had and which areas to avoid and identify various weaknesses in the platform we used."*

Manager 10 disagreed, "*we don't do it because Nick is in Johannesburg and I am here.*"

Manager 11 agreed, "*yes, but not formally. We discuss what went wrong or right but we don't do much of that. With our projects, its kind like an agile development methodology. We have daily scum meetings on ongoing basis but post, we usually look whether we have gone over budget. We may discuss where we feel things did not go right and where we can do things better in the future, but it is not really a detailed post-mortem.*"

Manager 12 said, "*no, but we should do that. Timing [is the problem]. We move to the next project and we are aware that we should be doing it but purely from a workload perspective, we don't make the time to do that but we should.*"

The results indicate that most organisations don't conduct full, formal post-mortem reviews although they do discuss progress during the development phase in meeting. This is worrying because the literature (Birk, Dingsøyr, & Stålhane, 2002; Dingsøyr, 2005) shows that they are the best knowledge management tools software organisations can utilise.

5.6. Knowledge management challenges

A further analysis of the responses found that companies also encounter some challenges as they implement knowledge management. The following statements indicating challenges were extracted from the responses:

Manager 4: *"First of all you must know what you are looking for? Finding information is more difficult and also learning from the information is difficult*".

Manager 5: "*I think one of the challenges is that the tools that we use change over time. In software development, when new tools come we have to learn them and apply them. So we have to move from wherever we were using before to that [new tools]. And as you get people into your team, each has got their own preferences as to how they want to document it [information]. It becomes difficult if you do not have a kind of a formalised system or somebody guiding that process quite formally. It is quite easy that information will be in*

different places and formats. Even if there is some sort of guidelines, the discipline to always put it in the right place and the people to appreciate it is the difficult part. Trying to get it into one place and getting it into people is difficult especially in a small company."

Manager 9: "*What is challenging is finding the correct resource quickly. For example, you want something specific and it is not necessarily documented. You have to search a bit and even ask other colleagues in the engineering space. The other challenge is that you have the knowledge and people don't use it."*

Manager 11 stated that the main challenge is time. He said, "*it's a time thing. We have a lot of pressure and we do not have the time. You just need the basics to get around.*"

Manager 12 explained: "*the biggest challenge we probably have is protecting our IP. In other words, how do you make it available to somebody to use and ensuring that nobody in the organisation takes it out? The second challenge is storage costs. If we store internally, the biggest challenge is the telecoms infrastructure which goes down from time to time and when it is down you don't have access to that information. If you move it to the cloud, you have associated costs in [United States] Dollars. Thirdly, the more you store the more you need. So often you have to go there and clean up because you have archived information in a place that might not be relevant."* The statements show the following knowledge management challenges that organisations face:

a) Lack of formal knowledge management procedures
b) Protecting the knowledge or IP
c) Storage costs
d) Never-ending need for information
e) Lack of time to fully adopt knowledge management practices
f) Difficulty in finding and learning from information
g) Ever-changing knowledge needs.

5.7. Summary of findings

Table 11 shows the summary of the results.

Table 11: Summary of findings

<table>
<tr><td colspan="3"><u>Knowledge gap experienced by organisations</u>
a) Knowledge of new projects
b) Knowledge of software development
c) Knowledge of new technologies
d) Knowledge about the industry they are developing for
e) Knowledge of clients</td></tr>
<tr><td colspan="3">Organisational knowledge management activities</td></tr>
<tr><td>Knowledge management process</td><td>Organisational knowledge management activities</td><td>Knowledge management technologies used</td></tr>
<tr><td>Acquisition</td><td>Acquiring knowledge from internal and external sources, e.g. Internet, clients, industry peers.</td><td>Internet</td></tr>
<tr><td>Creation</td><td>Reading and internalising information from different sources and applying it in tasks.</td><td>Internet and organisational databases/ knowledge repository software.</td></tr>
<tr><td>Storage</td><td>Storing acquired knowledge in databases and knowledge repositories.</td><td>Computer servers, personal computers, external hard drives, cloud.</td></tr>
<tr><td>Organisation</td><td>Placing knowledge in specific folders</td><td>System and specific application software.</td></tr>
<tr><td>Sharing</td><td>Meetings, brainstorming and training sessions and emailing.</td><td>E-mail, social networks</td></tr>
<tr><td>Application</td><td>Stored knowledge applied in software development, i.e. in installations, bug fixing and solving coding challenges.</td><td>Organisational databases and knowledge repository software.</td></tr>
<tr><td colspan="3"><u>Post-mortem reviews</u>
Are rarely conducted</td></tr>
<tr><td colspan="3"><u>Knowledge management benefits</u>
a) Time saving
b) Retain intellectual property
c) Effectiveness and efficiency
d) Knowledge reuse</td></tr>
<tr><td colspan="3"><u>Knowledge management challenges</u>
a) Lack of formal knowledge management procedures
b) Protecting the knowledge or IP
c) Storage costs
d) Never-ending need for information
e) Lack of time to fully adopt knowledge management practices
f) Difficulty in finding and learning from information
g) Ever-changing knowledge needs.</td></tr>
</table>

5.8. Responses of developers

Developers from the organisations included in this study were interviewed to investigate their knowledge management practices. They were interviewed because the researcher believed that for knowledge management to be adopted in the whole organisation, individual developers have to be actively involved. Sharma, Singh, and Goyal (2012, p. 24) are of the view that the success of organisations depends on the knowledge held by its employees. They further say that, in the software development environment, the success or failure of software development projects depends on the knowledge held by developers. "The software engineers are the face of the software engineering organisation and can reflect the true picture of the state of affairs in the area of knowledge management in the software industry" (Sharma, Singh & Goyal, 2012, p. 24).

Using the knowledge management lifecycle framework adopted in this study, developers were asked questions on how they acquire, create, store, organise, share/transfer, and apply knowledge. The interview responded to research questions five and six. The results are presented in the following subsections.

5.8.1. Knowledge need

The first question that developers were asked was whether they experience knowledge gaps or knowledge needs in their daily development tasks. For one to adopt knowledge management, there must be a knowledge gap that has to be filled. This question was aimed at identifying the developers' knowledge needs. When asked if they encountered knowledge needs, developers responded as follows:

Developer 1: "*Yes [I encounter knowledge needs]. I have been around here for a while but I am not really trained as a programmer."*

Developer 1 added, "*as part of the programming, we develop systems for other companies from other industries, so it is not only IT knowledge that one is looking for. Sometimes I have to learn about the other industries. Whether it is a financial company, I have to learn about it."*

Developer 2: "*From a developer point of view, you find gaps at times. I am not qualified but I have learned myself. Generally, I find gaps when I am coding."*

Developer 3: "*Well, within the IT field, there are many problems that you might encounter and you might not be fully equipped to take one, especially different [programming] languages and stuff. There are various languages you can develop or code in and if you are not taught or not good in that language, obviously you have to acquire that information in some way. There is a misconception that many people have about IT and I held that misconception too before joining this company. It is that IT is purely technical. What I realised was that as a programmer you also need to understand how business systems work."*

Developer 4: "*Yes. Usually if you want to create a program that you have not done before, you need to do research about it. That's it basically. For example, maybe if it is a search query for a database or connecting to an SQL server anything like that."*

Developer 5: "*It is more getting to know the business."*

Developer 6: "*Yes, it does happen. There is business knowledge and development [that I experience a gap in]."*

Developer 7: "*Yes, it happens every day of life*."

Developer 8: "*Sure! One thing about being a software developer is the nature of the [programming] languages, utilities, libraries and the frameworks that are coming out. The problem is that new tools and libraries come out quickly; you use one tool today and six*

months down the line that library is obsolete and you have to learn something else. There is constant need that you have to learn."

The results reveal the different knowledge gaps that developers face. There are two types of knowledge gaps that they have, namely development and business knowledge needs. One assumes that because of the nature of their work, they require only technical programming knowledge, yet they also need to know the industry they are developing for.

5.8.2. Knowledge acquisition

In order to establish how they addressed their knowledge needs, developers were asked to state the knowledge sources they used to fill their knowledge gaps. They revealed a number of sources that they used. When the question was posed to them, they responded as follows:

Developer 1: "*I use sources like the Internet and books and we have guys from other companies that come and train us or assist us if we have some issues."*

Developer 2: "*In that case, I go online or external sources. We have a partner company that we call for assistance and internally we do have documentation from previous developers that I usually use."*

Developer 3: "*The first and foremost resource that I use is the Internet. There are different kinds of links [that we use]. I also make use of senior colleagues because they know more than me. I look up to them for help whenever I need it. There are also some books here. I read a lot of books at the beginning of my employment here, and they helped greatly and I still refer to them."*

Developer 4: "*Well, either the Internet or the 'box' (a folder containing PDF files of programming related information) related to that gap. I look at the Internet, in the box or books."*

Developer 5: "*I have been developing for 25 years. I got most of the knowledge through studying. Google is a very important source of knowledge too. Sometimes you have to rely on other developers who have a different type of knowledge than you have for that particular project."*

Developer 6: "*If it is business knowledge, you get from clients and if it its development knowledge, you get it from websites or books. We also look at competitors' packages at times."*

Developer 7: "*We have a [online] user group with a knowledge base [a database]. If I do something out of the ordinary, I get information from the user group, but if I don't get the answer, I do it myself. There is also a local user group and we meet once a month. It's about ten of us talking computers and problems like that."*

Developer 8: "*You have to basically learn on your own time. We use the Internet, books and videos. For example, it could be online courses. In most cases we use books and online resources."*

The results show that developers acquire knowledge from a number of sources. According to the responses, the Internet is the main source of knowledge for developers. Other very important sources of knowledge as indicated by developers are books, organisational documents, other developers (from within or outside the organisation), clients and competitors. Competitors do not seem to be used much; only one developer indicted that they sometimes use their information.

5.8.3. Knowledge creation

Another way of satisfying a knowledge gap is to create your own knowledge. A number of knowledge creation activities and sources are used. Developers were asked to state how they create new knowledge and update existing knowledge to satisfy their knowledge needs. These are their responses:

Developer 1: "*What we do is that we have developed a system whereby if we are doing work for a specific client we create folders for the client in the server. If we encounter problems, we refer to that folder."*

Moreover, "*even for myself, if I work on something and I see an error that I have documented, I go back and check so that I can relearn that error myself. The head of the company encourages everyone to take time off and improve their skills set and develop themselves."*

Developer 2: "*I usually consult the sources I have mentioned earlier [Internet, documents, and clients]. Generally, the Internet, that's where I get most of the information."*

Developer 3: "*We usually have guys who are more qualified than us who come to teach us here." [also mentions sources mentioned earlier, Internet, books etc.]*

Developer 4: *[mentions processes and sources used for knowledge acquisition, for example, Internet, books etc.]*

Developer 5: "*With development, you literary have to work every day otherwise you are not growing with technology because it changes every day and so quickly. Courses are good and keeping documentation is also good, but I don't think it is the way to learn because you really learn when you are in a project and someone wants something and that is how you do it."*

Developer 6: *[mentions sources mentioned earlier, Internet, books etc.]*

Developer 7: "*The user group is very useful in letting me know what is going on that I have not heard about and I would follow it up. We mostly develop for Microsoft and they send notices to us."*

Developer 8: "*I use the sources that I have just mentioned [the Internet and books]. For example, if you are learning a new library, I would go to that library website and try to update myself with the documentation that they provide. If I have learned version 2.1 of that library, I will go back a month or two later because now we using version 2.3 and I would see those changes."*

Developers were further asked if they attend knowledge creation activities such as training workshops, seminars and conferences. The results indicate that they don't. In their own words they said:

Developer 1: "*No we don't."*

Developer 2: "*No, I haven't done any official training here but I remember a couple of years ago, I did go to a conference where there were IT specialists doing like a training or seminar on this thing [development]. We were exposed to it but we haven't taken up training ourselves."*

Developer 3: "*We are a small company and our software is not complicated. It is easily controllable, [but] we have had people coming here to train us, but it's something informal."*

Developer 4: "*I haven't attended one."*

Developer 5: "*Well it's something I have proposed and probably we will start in the new year. Every Friday afternoon, the teams in the different provinces get together and select a person for that week to give us a course on something they know and think that everybody else does not know. So we do a training session and everybody takes part."*

Developer 6: "*There is no formal training here."*

Developer 7: "*No, I think the user group is my main source of knowledge*."

Developer 8: "*Not really, Durban does not have these things compared to Cape Town and Johannesburg. The budget is low for such. I would like to go for these conferences in the United States and the United Kingdom."*

The developers' responses show that they use the same sources they use for knowledge acquisition for knowledge creation, although a few stated other activities such as on the job training. It is clear that developers are not exposed to knowledge creation activities such as formal training, conferences and workshops. Training is mostly informal.

5.8.4. Knowledge storage

After creating and acquiring knowledge from different sources, developers were asked if they store their knowledge and where they store it. The results reveal a number of storage devices that are used by developers. When asked if they store knowledge and how they store it, the developers said,

Developer 1: "*We run everything in the server and we can use it at any time we want."*

Developer 2: "*Yes, we do. We have a portal like space in the server where everyone can access documents but we also have bits of documents from our own side [our computers]."*

Developer 3: "*We usually keep it in our computers, in the server and on paper."*

Developer 4: "*I basically store everything in my computer and I also use an online journal."*

Developer 5: "*We have a central database, we store and download the source code from there."*

Developer 6: "*We have a knowledge base with files."*

Developer 7: "*It is all in the head."*

Developer 8: "*I store it partially because most of the time I remember these things. Maybe once in a while I will store some snippets of code in our Wiki page, but I do not really store anywhere. It is usually in the head."*

The results indicate that developers store their knowledge in three main storage devices: servers including knowledge databases, personal computers and their heads. This means that developers store knowledge mainly in the organisations' servers and personal computers.

5.8.5. Knowledge organisation

After storing the knowledge, it is important to organise it for easy retrieval. Developers sometimes work with large amounts of information, which makes its organisation very difficult yet so important if they want to easily access it. They were asked how they organise the knowledge resources they have. Their responses that,

Developer 1: "*We have a file system in the server. Every client has their own folder which contains relevant files and all the documentation that they need and I think that is pretty much what we do."*

Developer 2: "*I try my best to organise the information. For example, if it is something on coding, I will sort it out in the hard drive and if it is information about product files, I will try to keep it in its own place."*

Developer 3: "*We have a server where we keep this knowledge. We have folders that are broken down. For example, I have software folders and module folders where I can actually go there and find information about the different modules."*

Developer 4: "*If we have a program that we are working on, I will make a separate one that I will play around with. Whatever problems I come across in the real program, I will recreate in the dummy program."*

Developer 5: "*We have a planner and the planner has all the information about the developers. So he keeps all the projects that are being done and we are able to see at any stage where each developer is with their project and then we control and manage it."*

Developer 6: "*We have our knowledge base with articles where we deal with business processes and technical knowledge."*

Developer 7: "*I don't organise it, I wish I could try 30 years ago."*

Developer 8: "*It is usually organised in the Wiki but it is hardly organised because it is not usually stored."*

Developers organise their knowledge mostly by using folders either in their computers or in the company server. The folders contain specific information related to specific tasks. One developer revealed the use of a planner.

5.8.5. Knowledge sharing

Knowledge sharing is regarded as very important in the knowledge management process. The belief is that knowledge that is not shared is not useful in the organisation. Developers were asked to indicate how they share the knowledge that they have with others inside and outside the organisation. The results indicate that developers share knowledge in a number of ways. In their own words, they indicated that they share knowledge as follows:

Developer 1: "*Normally what happens when I have discovered something and what the others do, is that we generally help each other out. Generally speaking, if we find a solution*

to a problem we have all looked at, we just speak to each other and be like hey do you remember this error and this is how I solved it and you remember next time."

Developer 2: "*What I would do is that, I would collect my information and forward what we need to everyone else. I mostly use the Internet."*

Developer 3: "*We use the Internet by emailing each other useful links."*

Developer 4: "*I share with my boss because he is also on development. I would show him on my computer and say this is what I found. This is usually face-to-face."*

Developer 5: "*What we do is that every day we have a scrum meeting because were are based in [different locations]. We have a Facebook meeting and we call it a scrum and in the scrum everybody talks about their projects they are busy with and how far they are and show the code and everyone has the opportunity to view the code. Other developers start commenting on the code and in that, knowledge is being shared."*

Developer 6: "*We usually have daily meetings in the company where we discuss issues. Everyone is available even if they don't contribute but at least they know what we are talking about from a day-to-day basis. They have a picture of what we are doing."*

Developer 7 indicated that he shares the information with the two user groups; the online and the local user groups as stated earlier ("*There is also a local user group and we meet once a month. It's about ten of us talking computers and problems like that"*).

Developer 8: "*It is usually face-to-face because we sit next to each other. It is usually ad-hoc. If it is long, we use a Wiki. If someone is working from home, I would say hey, look here what I found."*

Others went further to indicate how they share knowledge with external stakeholders. This is clients, developers from other organisations and students.

Developer 1 said, "*We have all the information about our clients in our main system. if we want to bring it up with the client, we use it to do that.* He further stated that they also share knowledge with other companies in the industry. He said, '*if we can help we help if its IT related*".

When asked how they share knowledge with outside stakeholders, Developer 2 said, "*I would say it's quite a mix between assisting them online or organising a meeting onsite or having someone come here and spend time with us. With clients, we like assisting online or just log into their computer remotely and assist them, but if we feel that we need to travel onsite we will schedule sometime so that they can come and help*."

Developer 3 said, "*It is part of our policy to train clients to use our software. We usually train them onsite. [We] usually use the software while we teach the clients."*

Developer 5 indicated that they transfer information to interns from college. She said, "*we get people straight from college and we teach them so that they can have experience when they get the job*."

The results indicate that developers share knowledge within the organisation and with external stakeholders. They share business and technical (development) knowledge. In both instances, they use face-to-face interaction and technology. Through face-to-face meetings, they help each other if they solve technical problems, during meetings and training sessions.

5.8.6. Knowledge application

Another important process in knowledge management is knowledge application. Developers were asked to indicate how they apply the knowledge they have in their daily tasks. They all indicated that they do apply the knowledge in their development tasks. They were then asked to give examples or cases where they have applied the knowledge in their tasks. They responded as follows:

Developer 1: "*Well it depends on the situation. I just do trial and error [and find the solution]. Next time I borrow from that structure and the underlying planning and use that again. If it involves bug errors that once came out [that we dealt with], we document those bugs and make reference to them."*

Developer 3: "*I like filing everything I have learned. So every time I encounter a problem that requires that knowledge, I just sift it out and use it."*

Developer 4: "*Everything that we acquire we use in our development."*

Developer 6: "*I was doing a data warehouse, something I was not familiar with until recently. I had to do research and trial and error using the resources we have in the company*."

Developer 7 narrated a situation in which he used knowledge from their development network to solve a problem. He said, *"a couple of weeks ago, I had a difficult query with SQL. The whole think could not work. I called two gurus in our network and they helped me put it right."*

Developer 8: "*What I do the next developer knows. The information we would use if we hire someone and we assign them specifically to that project. To start off, we would advise them to go and look at the Wiki to get up-to-date with how you need to start off."*

It is clear that developers use the knowledge that they acquire, create, and store in their tasks. This is encouraging because it shows that they don't only acquire and store knowledge without applying it in their everyday tasks. The question now is: do they see the benefits of using such knowledge? The next section answers that question.

5.8.7. Knowledge management benefits

Developers were then asked to indicate if they have benefited or learned to do things better because of the knowledge management practices. The results show that knowledge management has indeed benefited them. Developers confirmed and said the following:

Developer 1: "*I can see a huge difference now than three years ago. Three years ago, there was no keeping of information about bugs and no one was keeping track of the webpages [we use] and definitely there was no documentation. Since I got back here and starting this, I believe that things have improved."*

Developer 2: "*I have been here for eight years. Previously, we did not document and knowledge was not shared at all. People had the knowledge but did not help everyone. We have addressed that and we have started putting documents together and with new staff coming in we have been showing them the knowledge locations."*

Developer 2 continued and gave an example in which knowledge management played a vital role in his development task. He noted,

"Three years ago we developed software and the installation was a failure the first time I went there because of issues [bugs]. Now we no longer get there because we have bugs checking procedure and all the testing which is done onsite everything is documented now."

Developer 3: "*I have grown immensely. I cannot even measure the growth I have acquired in this company both as an individual and learning business systems. I learn every day, every month, and throughout the year."*

He continued to narrate a story about how knowledge management has helped him. He continued and said,

"When I first got here, I used to tackle a problem, solve it and forget about the solution. Over time, I have learned to store my solutions and now I am more productive. If I get a problem, I can fix it faster because I know the solution at hand rather than going back and solve it again and get it back into my head. I can sift the solution and use it faster thus solving the problem much faster."

Developer 4: "*Yes, I look at how I used to code before when I started working here and compare it to now and I have looked at the projects that I have done before on my own and the knowledge that I have acquired. I can say that that I can see a lot of improvement on the projects that I did before with the knowledge that I have now. I feel that there is a huge improvement."*

Developer 5: "*Totally, I think what is more important is the business and developers' knowledge. If I put the two together, definitely they have helped me in my job. For example, if you look at Google, there are a lot of forums we did not have five years ago. You would ask or talk to no one if you had a problem, now everyone is on the forums and you can ask questions on Google and people will reply on these forums. Most definitely it has improved my work hugely."*

Developer 6: "*I think so, currently we tend to use agile development methods as opposed to traditional methods. We tend to be high level and respond to issues better and quicker and*

respond easier to customers' needs and changes to customers' needs. So I think it is better now."

Developer 7: "*No, I don't think so. Personally, I have not become efficient. I think the software has improved over time which has made me more efficient. For example, something that took me a week to do can take me a few hours now because of the software."*

Developer 8: "*It saves time. If you think about it, I don't have to repeat something I know. Every project is a little bit different so I don't have to explain for five, ten, minutes of my time, that's the good thing about documenting your knowledge, you don't have to repeat yourself."*

The results indicate that developers' work has been greatly improved by knowledge management practices. They have stated that their work is more efficient and they are more productive as compared to previous years. Only one developer indicated that this was due to technological advances.

5.8.8. Summary findings of developers

Table 12 provides a summary of the developers' knowledge management practices.

Table 12: Summary of findings of developers

<table>
<tr><td colspan="3"><u>Knowledge gap</u>
a) Development knowledge
b) Industry knowledge</td></tr>
<tr><td colspan="3"><u>Knowledge management activities by developers</u></td></tr>
<tr><td>Knowledge management process</td><td>Developers' knowledge management activities</td><td>Knowledge management technologies used by developers</td></tr>
<tr><td>Acquisition</td><td>a) Reading books and other organisational document
b) Attending training
c) Getting help from industry peers</td><td>d) Internet
e) Company knowledge databases</td></tr>
<tr><td>Creation</td><td>a) Internalisation of information from different sources
b) Sharing ideas with development groups members</td><td>c) Internet
d) Company knowledge databases</td></tr>
<tr><td>Storage</td><td>Storing in company servers and personal computers</td><td>Computer technology</td></tr>
<tr><td>Organisation</td><td>a) Organised in Wikis, folders
b) Using a planner</td><td>a) Wikis
b) Electronic planner</td></tr>
<tr><td>Sharing</td><td>a) Helping each other through face-to-face interaction
b) Meetings
c) User groups
d) Training
e) Sending each other information</td><td>Internet</td></tr>
<tr><td>Application</td><td>Applying knowledge in development and in new employee orientation.</td><td>Not applicable</td></tr>
<tr><td colspan="3"><u>Knowledge management benefits</u>
a) Saves time
b) Improved efficiency and effectiveness
c) Improved performance</td></tr>
</table>

5.9. Summary

This chapter presented the results of the study. Three data sources were used: software development managers and developers. Results from software development managers indicate that organisations face software development failures and that most of these failures fall into two types of failure: abandoned projects or challenges. There are ten causes of these failures: bureaucracy in IT departments, compatibility issues, complacency of developers, involvement of wrong people at the planning stages of the project, lack of detailed documentation, lack of resources, lack of user commitment, miscommunication, unrealistic customer expectations, and work overload. Further to that, the results indicate that software development companies practice knowledge management. These practices are in accordance with the conceptual and theoretical framework of the study. That is, software organisations acquire, create, share, organise, store and apply their knowledge in the development of software. These knowledge management practices have benefits to organisations. The main benefits are time saving, retention of intellectual property, efficiency and effectiveness and knowledge reuse. As much as there are benefits, there are challenges too. The main challenges are that in most organisations there is a lack of formal knowledge management procedures, protecting the knowledge or intellectual property (knowledge) is difficult and expensive, there is a never-ending need for information, difficulty in finding and learning from information, and ever-changing knowledge needs. Developers' responses are almost the same as those of their managers. This would mean that they complemented their managers' responses.

CHAPTER SIX
DISCUSSIONS OF THE FINDINGS

6.1. Introduction

This study sought to understand the nature of software development failures experienced by software development SMMEs in South Africa with the view of investigating how knowledge management could be used to alleviate them. Specifically, the study answered these questions:

1. What is the form and source of software development failures experienced by software development SMMEs?
2. What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
3. What benefits has knowledge management brought to software developers and organisations?
4. What knowledge management challenges do software development SMMEs face?

6.2. Software failures in software organisations

The study found that software development failures are prevalent in South African software development SMMEs. In this study, all but two organisations indicated that failures are prevalent in their respective organisations. The results indicate that this issue is persisting in South Africa and globally, confirming findings of other studies carried out earlier in South Africa and globally. In South Africa, results of latest ITWeb surveys (2011, 2012, 2013) indicate that only 11% of software development projects (in-house or outsourced) are absolutely successful (that is, they are delivered on time, on budget and with required functionalities). Two percent are a complete disaster (in most cases abandoned or never used) and the rest (87%) are challenged (usually over budget, late, and with less than the required

functionalities). The latest South African study on software failures by De Wet and Visser (2013) confirms previous studies. Their study concluded that the success rate of software projects in South Africa is low, just like in other parts of the world (p.26). Previous studies by Smith (2002) and Labuschagne and Marnewick (2008) also indicate the same state of affairs. Smith (2002) concluded that failure rates in South Africa were lower than in the rest of the world, yet the failure rates were way over 50%. Even though the conclusion was that they were low, 50% is very high to consider low. That means, the failure rate is very high even in South Africa. Labuschagne and Marnewick (2008) found that 27% of projects failed and 36% were challenged, making a total of 63% of projects that are outright not successful in South Africa.

Research in many other developing countries like South Africa has also reported high information systems failure rates. Nauman, Aziz and Ishaq (2005) state that less than 40% of software projects in developing countries are successful. Heeks (2008) reports that, in developing countries, 35% of e-government projects are total failures and 50% are partial failures, leaving only a 15% success rate. Rahman (n.d.) shares the same sentiments and states that only a few projects are absolutely successful in developing countries. Tarawneh, Al-Tarawneh and Elsheikh (2008) also confirm that and state that in Jordan there is a very high failure rate of software development projects.

Globally, the Standish Group's reports (2013, 2014, 2015), El Emam and Koru (2008), and Cerpa and Verna (2009) confirm the prevalence of software development failures. As indicated in the literature review, the Standish Group's reports from 2011 to 2013, as cited by Hastie and Wojewoda (2015), indicate that less than 32% of all software development projects are successful. This means that the majority are either a complete failure or challenged. In their study, El Emam and Koru (2008) found a cancellation rate of 23 % over a two-year period for projects. Jørgensen (2014) developed a model based on existing data and

predicted that 74% of projects fail. Kaur and Sengupta (2011, p. 2-3) reported statistics of a number of publications on project failures. Their report indicates that less than 40% of projects in developed countries fail. The results of the study and the analysis of the literature indicate that software development failures are prevalent in South Africa, just like in other developing and developed countries. This shows the seriousness of this issue in the software development industry. The results of this study and the literature shows an industry in serious trouble.

6.2.1. Types of failure

The study also determined the types of failures that software organisations experience. Failures are classified differently by different authors and scholars as discussed in detail in Chapters 2 and 3. This study adopted the Standish Group's classification of failure. That is, projects either fail outright and are abandoned or they are challenged (the Standish Group's CHAOS Manifesto, 2013, p.1). The results of the study indicate that South African software development SMMEs experience both types of project failure. Projects either fail outright and they are abandoned or they are challenged. The results show that the organisations have experienced mainly total failures which has led to projects being abandoned at some stage. This is contrary to most studies in the literature. The ITWeb surveys (2011, 2012, 2013) showed a lower outright failure rate than challenged projects. The Standish Group's reports, as cited in Hastie and Wojewoda (2015) also show a similar trend. Even academic studies (El Emam & Koru, 2008; Cerpa & Verna, 2009) show a lower outright failure rate than challenged projects. These findings are influenced by the research methodology. Most of the studies conducted on this issue adopt a quantitative methodology. This study adopted a qualitative methodology. As much as the results are a true reflection of the study, readers are urged to accept them with caution because they are contrary to many studies conducted previously.

6.2.2. Causes of failure

Ten causes of software development failure (outright failure or challenged development) were discovered in this study. They are:

- Complacency of developers
- Compatibility issues
- Unrealistic customer expectations
- Bureaucracy in IT departments
- Lack of user commitment/non-adoption
- Lack of detailed documentation
- Lack of resources (financial and human)
- Miscommunication/misinterpretation of requirements
- Work overload
- Involvement of wrong people in the projects' planning stages of projects.

Eight causes of failure (complacency of developers, compatibility issues, unrealistic customer expectations, lack of user commitment, lack of documentation, lack of resources, miscommunication, and work overload) appear repeatedly in the literature and two (bureaucracy in IT departments, and involvement of the wrong people in the planning stages) seem to be unique to the South African context.

The issue of unrealistic customer expectations has been appearing in the literature for a long time. Charette (2005, p. 45) discusses this issue and concludes that it is the main cause of software project failures. In most cases, the customer expects some kind of utopian software that can perform extremely complex tasks, which leads to disagreements between the client and the developing organisation about the quality of the product when it is delivered. One

respondent indicated that such a situation affected his company and he had to walk away from three clients. For three consecutive years, the ITWeb (2011, 2012, 2013) and the Standish Group (2013, 2014, 2015) listed this issue as a cause of failure. It is also found in Smith (2002) and in many other studies. As much as most of the time it is the customer that is guilty of unrealistic expectations, the study found that software development companies sometimes promise what they cannot deliver. They promise the customer that the product will have functionalities that are capable of helping them improve their businesses, but find that the development companies have not been able to achieve that. This is confirmed by Bupa (2005, p.15) who states that the problem is caused by "suppliers over-promising on what can be delivered to win contracts".

The conclusion that is drawn from this is that both customers and software development organisations can cause software projects to fail because of unrealistic expectations. In this study, it was found that unrealistic expectations are associated with the quality of the end product, but it is also assumed that they are in the form of unrealistic timelines and low budget. In such circumstances, clients and management set very tight deadlines for project completion and approve a low budget to save costs. Companies usually don't challenge tight time schedules because they fear losing business and start developing under tight deadlines and low budgets, thus compromising the successful completion of the project. The issue of the importance of time is highlighted by Kaur and Sengupta (2011, p. 3) who state that time is important in software development and it must be well calculated because it affects the project development.

The non-adoption of systems or lack of user involvement has been a thorny issue in organisations and a subject of research by academics. Sometimes it happens that systems are developed and have perfect functionalities but are rejected by the user as managers indicated in this study. Tarawneh, Al-Tarawneh and Elsheikh (2008, p. 249) ranked this failure as

cause number two in their study in Jordan, and they suggested that users must be involved in the design of the system to eliminate vagueness in specific items of the work which the development team alone cannot identify. Geethalakshmi, and Shanmugam (2008) concur and state that the level of user or customer involvement contributes more to project success or failure and they suggest that users must be involved at all stages of the project. Kaur and Sengupta (2011) are of the view that lack of user commitment occurs when no one is committed to a system and people are hostile to it. Research has found many reasons that cause users to resist systems. Davis (1989) explains that users adopt systems because of ease of use and perceived usefulness.

Venkatesh, Morris, Davis and Davis (2003) concur with Davis (1989) but include other factors such as effort expectancy, performance expectancy, social factors, facilitating conditions, gender, age, experience, and voluntariness as affecting user adoption. Rogers (2003) is of the view that innovations are adopted because of the nature of the innovation, how the innovation is communicated, time factors and the social system context. Software engineering literature indicates that systems are resisted for many reasons. Among them are hostile organisational cultures, if users are not satisfied with the system, and if the system threatens jobs (Ewusi-Mensah & Przasnyski,1991; Poon & Wagner, 2001; Yeo, 2002). To avoid this problem,

May (1998) is of the opinion that the interaction between users and developers must be encouraged to avoid the 'not invented here' syndrome which leads to users rejecting the system. McLeod and McDonell (2011) advise that users' expectations, attitudes and involvement have to be considered when developing or deploying a system. Rajkumar and Alagarsamy (2013) suggest that senior management must create a friendly environment in which the users of the system can actively participate in the project. In this study, there were no clear reasons why clients did not adopt the systems. The assumption is that the reasons are

the same as given in the literature, or it is the 'shifting of the goal post' as mentioned by respondents. This is an issue that needs further investigation.

Miscommunication and/or misinterpretation of requirements at the beginning and/or during the course of the development process has been identified as another cause of software development failure. When requirements are miscommunicated or misinterpreted, the systems might lack certain functionalities. A number of miscommunication scenarios were given by the respondents which led to the failure of systems during development. McLeod and McDonell (2011) state that communication is key if a system is to be successful. Unfortunately, miscommunication happens sometimes between projects stakeholders. It could be a lack of communication between team members or clients. Lehtinen *et al*. (2014, p. 627) state that sometimes developers are assumed to have understood ambiguous specifications. This is interpreted as a miscommunication between the developer and the client. May (1998, p. 10) suggests the establishment of a reasonable stable baseline requirements before any other work is done, but also cautions that "even when this is done, requirements will still continue to creep."

Kappelman, McKeeman and Zhang (2006, p. 35) are of the view that, if all those involved do not communicate and work together continuously, the project team will be pulled in different directions. If consensus is lost, there is little hope for the successful completion of the project. Maglyas (2010, p. 61) is of the view that "lack of effective communication decreases success chances significantly". McManus and Wood-Harper (2007, p.39) state that a lack of communication between project stakeholders leads to crucial business and technical knowledge not flowing well within the project. They suggest that there must be constant communication between clients and development teams during the development phase to avoid problems. El Emam and Koru (2008, p. 249) and Rahman (n.d.) add their voices to the debate and state that ineffective communication is one of the main causes of failure.

Managers also mentioned that experienced developers become complacent over time. This could mean that experienced developers might become careless and treat all development projects the same, yet projects differ in scope and size. Such behaviour will definitely lead to failure because mistakes are bound to happen. Harraf, Soltwisch, and Talbott (2016, p. 387) state that complacency can occur in four areas: processes, people, structure, and culture, and it is caused by a number of internal and external factors. In this study, it is clear that complacency is caused by experience, which leads to over-confidence and ignorance. McLeod and McDonell (2011) are of the view that developers can influence the success or failure of a project because of their skills and experience. In this case/study, experience led to failure.

The study found that, in some instances, there are compatibility issues between software products, which then leads to failure. In this study, it was found that it is compatibility issues between the operating system and the application software (the newly developed software) which led to a failure. The literature indicated that this problem is similar to other technological challenges such as over-engineered code and bad architecture (Smith, 2002; El Emam & Koru, 2008; Labuschagne, Marnewick & Jakovljevic, 2008). McLeod and McDonell (2011) are of the opinion that inappropriate technology selection and use can lead to challenged projects. In this study, it was the selection of inappropriate software (Linux) that caused the failure.

Bureaucracy in IT departments was found to be another cause of failure. This cause seems to be unique to South African companies because the literature does not mention it. The general opinion is that bureaucracy is responsible for delays. It is assumed that bureaucracy entails too much rigidity in the allocation of resources to projects, which then leads to delays. Such bureaucratic tendencies can obtrude at any stage of the development process. Bureaucracy

can also be a stumbling block if project teams are supposed to follow long protocols in term of effecting changes to projects or simply moving resources around.

Involvement of the wrong people in the project planning stages was also revealed to be a cause of failure. In some cases, customers who have no development knowledge get involved in the planning stages of projects. In such cases, the belief is that powerful people (it could be clients or top management) interfere in the planning stages of the project. They impose themselves yet they have no knowledge of the task. In the process, the development team is put under pressure and start developing without even being sure what they are developing. This will definitely lead to the project failing. This cause is also unique to South African companies because it has not been mentioned in the literature from elsewhere.

Work overload can also cause failure. Developers can be overloaded when the time left to finish the project is not enough or when one or a few people are doing the task. Kappelman McKeeman and Zhang (2006) discussed this issue and indicated that if not enough time is allocated to project teams, that will definitely lead to failure. Bupa (2005) also confirms that a shortage of staff can lead to failure. Rahman (n.d.) is of the view that staffing is one of the most important elements in the success of a project. Without enough staff there is no guarantee of project success.

Lack of resources (financial and human) is confirmed by Lindberg (1999) who states that a lack of resources can affect project success. This could refer to financial and/or human resources. Human resources means skilled developers and project managers. Lindberg (1999) gives an example in which skilled people start the project late and software teams compete for skills against each other. Kappelman, McKeeman and Zhang (2006) also add to this debate. They state that the lack of human skills causes failure. It has been indicated in the literature that lack of skills can be a lack of technical or human skills. For example, lack of

project management skills has come out strongly as a cause of project failure. Small software development organisations also lack the financial skills that big organisations have. McManus and Wood-Harper (2007, p. 39) state that "few organisations have the infrastructure, education, training, or management discipline to bring projects to successful completion". When organisations encounter cost overruns or time delays, more financial resources are supposed to be committed to the project. If the organisation lacks such resources, it becomes difficult to continue with the project. This will definitely cause the cancelation or abandonment of the project. Rahman (n.d.) is of the view that an insufficient budget is a major reason for missing the goals and objectives of a project. Ebad (2015, p.1150) offers a different opinion about cost and asserts that they have no clear direct impact on failure. Judging from the literature, the conclusion is that costs do have a direct impact on failure.

Respondents indicated that a lack of documentation during the specification stages of the projects causes problems because those involved tend to forget or misinterpret what the customer actually wants and develop something completely different from the actual specifications. Kappelman, McKeeman and Zhang (2006) state that if development processes are not documented, project team members and stakeholders will have different views of the project thus making them form different perspectives.

6.2.3. Comparison to other developing countries

The results of this study were then compared with results from other developing countries to determine differences and similarities of the findings. This was done to determine if there are common or different trends in the developing world. The comparison aimed at creating some kind of a bench mark for organisations. A few studies have been identified in the literature for this comparison. In Pakistan, Nauman, Aziz and Ishaq's (2005) found that failures are caused by non-adoption of the system by users, undefined organisational processes at project

inception, and ever-changing changing specifications. In Jordan, Tarawneh, Al-Tarawneh and Elsheikh (2008) found that the user's misunderstanding of requirements, user involvement, executive support, clear business objectives, minimum size/scope of the project, reliable estimator factor, and manager factors could lead to success. This means that if not implemented they lead to failure. In India, Geethalakshmi and Shanmugam (2008) found that the level of user involvement, software process management, estimation and schedule contribute most to project success and failure. Maglyas' (2010) Master's dissertation identifies development processes, requirements management, architecture design, and resource estimation as major challenges to project success in Russia, Ukraine, and Belarus. In Nigeria,

Egbokhare (2014) found that isolation of users from the development team throughout the development process, vague customer requirements, lack of experienced developers due to lack of financial rewards, lack of documentation from past projects, poor maintenance culture, slow response to changing technology and organisational politics are causes of failure. Ebad (2016) says that in Saudi Arabia, failures are caused by a lack of top management support, organisational culture, lack of training, business process reengineering and unavailability of project management offices. On the other hand, Vithana, Fernando, and Kapurubandara (2015) found that decision time, customer satisfaction, customer commitment, corporate culture, training and learning, team size, and technical competency are factors that lead to project success in Sri Lankan agile teams. Table 13 summarises the results.

Table 13: Comparison of project failures

Causes of software development failures found in this study	Common causes of software development failures found in other developing countries
1. Bureaucracy in IT departments 2. Compatibility issues 3. Complacency of developers 4. Involvements of wrong people in the planning stages of the project 5. Lack of detailed documentation 6. Lack of resources 7. Lack of user commitment/non-adoption of systems, 8. Miscommunication and misrepresentation of requirements 9. Unrealistic customer expectations 10. Work overload	1. Badly defined requirements 2. End-user resistance 3. IT related issues 4. Lack of top management support 5. Lack of training 6. Poor project management 7. Unrealistic or unarticulated goals

The comparison shows many similar causes of failure within the developing world. This study found only three more from the rest of the other countries, but this does not mean that these differences are too glaring to be considered. The conclusion drawn based on this comparison is that the failures are the same, with minimal differences which are insignificant.

6.3. Knowledge management practices at individual and organisational levels

After determining the software development failures, knowledge management practices of individual software developers and the whole organisation were then investigated. This was done because of the belief that knowledge management at individual level influences organisation-wide practices (Argyris & Schon, 1978, 1996). By determining knowledge management practices the study sought to determine the role that knowledge management plays in reducing or eliminating failures. First, the knowledge needs of developers and the whole organisation were determined.

6.3.1. Knowledge need

The results of the study indicate that software development organisations experience knowledge gaps or knowledge needs. There are two types of knowledge needs that the organisations have: development/technical and business knowledge needs (*"There is business knowledge and there is also development knowledge that we usually seek"*). In this study technical knowledge was found to be programming or coding knowledge; that is, knowledge about the technical development aspects of the software development process, for example, programming a database query. Tiwana (2004a, p. 900) lists design knowledge (design patterns, design models), programming (programming languages, and tools), and software processes (methodology and debugging procedures) as technical knowledge. Business knowledge is all about the software industry and the industries they are developing programs for. For example, if they are developing for a financial company, the software development company must know about the financial sector. They should also know about their own software development industry; that is, the industry trends, latest developments and their competitors.

Tiwana (2004a, p. 900) states that business knowledge is all about business processes and customers' objectives for the software product. These results are consistent with results that have been found in other studies across the globe. Mishra and Mahanty (2015) found that software projects require high levels of business knowledge. They divided business knowledge into domain, regulatory, strategic, process, and operational process knowledge (p. 83-84). Based on these classifications, the study found that domain, regulatory and operational knowledge is the most important in software organisations. As presented in the previous chapter, organisations look for regulations for the industry they are developing for (in this study, tax and electrical standards). They also look for domain knowledge; that is, knowledge of the industry they are developing for and operational knowledge; that is, general

knowledge about running the daily operations of the business. Curtis *et al*. (as cited in Walz, Elam & Curtis, 1993, p. 63) agree and state that multiple technical and domain specific knowledge is required for most complex software development projects. Lindvall, Rus, and Sinha (2002) echo the same sentiments and divide the knowledge needs of software organisations into technical and business. According to Lindvall, Rus, and Sinha (2002, p. 27) technical knowledge is knowledge about new technologies. They need this knowledge because new technologies can have positive and negative consequences in organisations. As a positive consequence, new technologies can help organisations perform better but they can also cause delays as developers take time to learn them. This is true of new software development languages and tools. They also emphasise the need for domain and regulatory knowledge. They state that organisations must acquire knowledge about the industry they are developing for and that organisations must make available policies and other regulatory knowledge to their employees, especially new employees. Tiwana (2004b, p. 51) concurs with the others and states that effective software development needs knowledge congruence; that is, a good mix of technical and business knowledge across the client vendor dyad. This means that the software development organisation must have sound knowledge of the client's environment in which the software is developed and that they should have the technical expertise to develop the required knowledge. Robillard (1999) takes a cognitive perspective of the knowledge needs of software development.

Robillard (1999) states that software development needs topic and episodic knowledge. Topic knowledge is written knowledge. Episodic knowledge is the experience held by an individual (p.88). These two types of knowledge are interpreted as codified knowledge and tacit knowledge held by the organisation. These two types of knowledge are technical and domain. Tiwana (2004a) suggests that the two types of knowledge should be integrated in the organisation because the integration is central to the software development process and it

increases effectiveness, reduces defects and increases efficiency. The conclusion drawn from this discussion is that South African software development SMMEs experience two types of knowledge gaps: technical and business knowledge. Their knowledge needs are not different from those of their global counterparts.

The knowledge needs of software developers were also investigated to determine if there is a difference between their individual knowledge needs and organisational knowledge needs. Walz Elam and Curtis (1993, p. 63) are of the view that knowledge is the raw material of software design teams, but often individual software team members do not have all the knowledge required for a project and must get additional knowledge before they can do their work. The study found that developers require technical and business knowledge. This is the same knowledge that managers indicated is needed by the whole organisation. Surakka's (2007) survey of software developers' skills revealed a number of ideal technical skills needed by software developers. Among them are programming, design, testing, documenting and project management skills (p. 75).

Citing literature from various sources, Liebenberg, Huisman, and Mentz (2014, p. 2605) concluded that leadership and negotiation, and project management are core skills that are needed and that knowledge in the business environment, consultancy and end-user computing are knowledge gaps and skills experienced by developers. Lethbridge (2000, p. 50) found ten knowledge and skills gaps experienced by IT professional. The gaps are a combination of business and technical skills. Examples are leadership and management (business knowledge), requirements gathering and analysis, real-time systems design and software metric (technical). Lee and Han (2008, p. 24) state that development, software, social skills, and business skills are highly required for programmers. Gallagher *et al*. (2011, p. 5) found that technical and business skills are required for IT professionals at entry level. Looking at the findings of the study and the literature, it is evident that software developers experience

knowledge gaps. The conclusion drawn is that software developers in South Africa experience technical and business knowledge gaps similar to those of their organisations.

6.3.2. Knowledge acquisition

After the knowledge needs of individuals and organisations were identified, the question that followed was: where do they get the knowledge to satisfy their needs? This study found that software development organisations acquire knowledge from a variety of sources. Knowledge is acquired from internal and external sources. Internally, they acquire it from each other and from organisational documents; externally, they acquire it from the Internet, clients and industry networks. The Internet was reported to be the main source of external knowledge. Gendreau and Robillard (2013, p. 2) agree that in software organisations knowledge is acquired internally and externally. They state that external sources are the Web, technical documentation, a book or paper. They further state that internal knowledge is the tacit knowledge in the developers' heads. That is, employees can tap that knowledge from each other. Studies conducted in other software organisations across the world found similar results.

Aurum, Daneshgar, and Ward (2008) found that in Australian software organisations, knowledge is acquired internally from informal networks of peers. They also found that organisations acquire knowledge externally from third parties such as experts and the Internet. Just as in this study, they reported that the Internet is widely used for knowledge acquisition (p.524). Trimble (2000, p. 175-178) presents five knowledge sources in software development. They are: collecting software metrics, communication, creating approaches to meetings, conducting observation sessions, and knowledge elicitation from experts. Their findings are similar to those of this study and the other studies encountered in the literature. This confirms that knowledge is acquired from internal and external sources.

Pástor, Šipikal, and Rehák (2013) state that in Slovakian IT companies, knowledge is acquired through company acquisitions. This allows less knowledgeable companies to tap into the knowledge of large knowledgeable companies. Employees are the biggest beneficiaries because intra-organisational knowledge is acquired and shared. Heavin and Adam (2012) say that software development SMMEs acquire knowledge from the following sources: people, courses, outsourcing, cooperation between source and recipient and documents. In one of their case companies, they found that knowledge is acquired from consultants, employee training, books, journals, conferences and from parent companies. Sharma, Singh and Goyal (2012, p. 27) state that knowledge is obtained mostly from corporate knowledge repositories, outside sources and co-workers. They also indicate that the Internet is widely used as an external source of knowledge.

Apart from software development organisations, other firms in general acquire knowledge from almost the same sources as software organisations. For example, Liao and Marsillac (2015) state that organisations can acquire knowledge from partners or strategic alliances, suppliers and informal networking. Ma and Huang (2016) are of the opinion that knowledge acquisition is essential because organisations lag behind without knowledge. They further state that organisations need market and technical knowledge and found that they acquire that knowledge from outsourcing partners. Holsapple, Jones and Leonard (2015) proposed ten knowledge acquisition processes which they divided into direct and indirect processes. Two are applicable in this study: reviewing professional literature and external knowledge acquisition. Akhavan and Dehghani (2015) proposed six knowledge acquisition techniques: interviews, observation, protocol analysis, laddering, concept sorting, repertory grid and mapping techniques. None of the six were found to be applicable in this study.

The results of this study and the literature indicate that software organisations acquire knowledge mostly from external sources, especially the Internet and external experts. Internal sources such as peers and documents are also used, but to a lesser extent. The conclusion drawn from this discussion is that the knowledge sources used by South African software development SMMEs are not different from the ones used by other software organisations worldwide. This is true for the use of the Internet and external experts as the main source of external knowledge.

After determining the knowledge needs of the whole organisation, the study sought to establish developers' knowledge sources. The study found that developers acquire knowledge from internal and external sources, just like their respective organisations. External sources are the Internet, clients, colleagues from the same industry and competitors. Internal sources are organisational documents (books, and organisational repositories) and colleagues. The Internet is the main source of external knowledge for developers too. This is not surprising at all, because the literature confirms it. Rezende and Alves (n.d.) found that the Internet is widely used by developers for knowledge acquisition. Sharma, Singh, and Goyal (2012, p. 27) concur and state that the majority of developers prefer the Internet as an external source of information to solve their problems. Vasanthapriyan (2015) also mentions Internet technologies such as forums, eLearning platforms, and knowledge portals as tools that developers use for knowledge management purposes. Apart from the Internet, Rezende and Alves (n.d.), and Sharma, Singh and Goyal (2012) mention that developers also acquire knowledge from organisational documents, vendors, and colleagues. The discussions reveal that the Internet is the main source of external knowledge for developers and that knowledge is also obtained from clients, vendors, organisational databases and colleagues within and outside the organisation.

6.3.4. Knowledge creation

Knowledge creation is the creation of new knowledge from available information resources (Pentland, 1995). Holsapple and Singh (2001, p.84) define knowledge-creation as "an activity that produces knowledge by discovering it or deriving it from existing knowledge". In this study, it was discovered that software organisations create knowledge by reading, brainstorming, and consulting experts. Knowledge is created when they internalise (as explained by Nonaka & Takeuchi, 1995) the information that they read, share and observe into new knowledge. Nonaka and Takeuchi (1995) emphasise the importance of knowledge creation in organisations. They provide four modes of knowledge creation activities: socialisation, externalisation, combination and internalisation. In this study, internalisation and socialisation are widely used knowledge creation methods. When employees read information from the Internet, organisational documents, and other sources, and understand and internalise it and then apply it in their tasks, it is equivalent to internalisation. When they share and refine ideas during brainstorming sessions and meetings and when they consult experts and then act on that refined knowledge, that is socialisation.

In software development studies, similar results have been found. For example, Spohrer *et al*. (2013, p. 5) discovered that knowledge is created by exchanging code, solution and opinions, and developers observing and sharing information with their peers, thus learning from them. Judging from these activities and comparing them with the findings of this study, one concludes that they happen during brainstorming sessions and meetings. Dorairaj *et al*. (2012) found that brainstorming, communities of practice, and self-learning, are common knowledge creation methods in software organisations.

At individual level, developers indicated that they do the same activities. Neves *et al*. (2011) found two software creation activities in software development, i.e. comprehensive software documentation and responding to change. Organisational documents as indicated in this study

are used for knowledge creation. They are read and internalised by developers, thus creating their own knowledge which is then infused into the whole organisation. Shongwe (2015) found similar knowledge creation activities in a higher education environment. He found that students who were developing software used the Internet and interaction with senior students and professional developers to create knowledge (Shongwe, 2015, p. 7).

Apart from the knowledge activities of the individual organisations and developers, the software development process itself is regarded as a knowledge creation process (Bailin 1997). Arent and Norbjerg (2000) state that SPI as a software development methodology influences the creation of tacit and explicit knowledge. Tacit knowledge is created through the sharing of ideas and explicit knowledge is created by the documentation of ideas. Morner and von Krogh (2009) share similar views and state that in software development, knowledge is created by the interplay between tacit and explicit knowledge. This happens when developers share ideas and use software development documentation. Wan *et al*. (2010, p. 494) concluded that support from top management, friendly project environment, and clear project plans play a huge role in knowledge creation. Lee and Cole (2003) took a different direction and investigated knowledge creation outside the software organisation. Their study was based on the open-source community. They found that knowledge is created by learning through criticism and self-correction.

Another knowledge creation method widely used in organisations is formal training. In this study it was found that it is not popular with software development organisations in South Africa. A few companies send their developers for formal training. This is contrary to the literature which states that training is a widely adopted knowledge creation activity in software development organisations (Kumarawadu, 2008; Sharma, Singh, & Goyal 2012). Kess and Haapasalo (2002) encourage training of all software development professionals. Dorairaj and Malik (2012) found that training is conducted in software development

organisations. The assumption is that small organisations might lack the resources for formal training activities; in fact, some managers indicated that precisely because of their small company sizes, they cannot afford formal training ("*look, we are small, sending people for training is not an option*").

The conclusion drawn from this discussion is that South African software development SMMEs and developers create knowledge in a similar way to their global counterparts. What is noticeable is that most studies indicate that the internalisation and socialisation are widely used knowledge creation activities in the organisations.

6.3.5. Knowledge sharing

Knowledge sharing is defined by Schwartz (as cited in Paulin & Suneson, 2012, p. 83) as the process of exchanging knowledge between individuals, and within and among teams, organisational depertments, and organisations. Ezeh (2013, p. 3) says that the knowledge stock of the organisation that is held by individuals within the organisation has great potential if shared but can also be a great barrier to software development if not properly managed (Carlile, as cited in Xu & Yao, 2013, p. 62). The results of this study indicate that in software development organisations, knowledge is shared internally and externally and that technology plays a huge role. Externally, it is shared with clients. Internally, it is shared with colleagues through face-to-face interaction, email and social networks. Face-to-face interaction usually happens during meetings, brainstorming sessions and informal one-on-one training sessions. The findings of the study are consistent with findings by Boden, Nett and Wulf (2010, p. 64) who found that software organisations shared knowledge by holding informal training sessions and meetings. The same results were found in Kuusinen *et al*. (2017, p. 141) who state that agile organisations share knowledge informally or formally in meetings. Informal meetings are usually between peers or colleagues and the formal meetings are usually used for sharing knowledge with customers. The same results are found in Aurum, Daneshgar, and

Ward (2008, p. 527), who state that in small software organisations, knowledge is usually shared among colleagues. Ezeh (2013) goes even further to explain the knowledge sharing facilitating factors in software organisations. Among them are physical location of colleagues, social relations, networks, meeting and formal spaces. These have been selected because they relate to the findings of the study. For example, managers indicated that sharing happens during meetings and during face-to-face interactions among colleagues.

Knowledge sharing practices in other organisations in general reveal a number of knowledge sharing practices, some which are similar to software organisations and some of which are different. For example, Rathi, Given and Forcier (2014) found that non-profit organisations share knowledge by adopting partnerships and that social networks such as Twitter are widely used as knowledge sharing tools. Aljuwaiber (2016) found that communities of practice play a vital role in knowledge sharing in organisations.

In this study, it was revealed that developers share knowledge within the organisation and with external stakeholders. They share business and technical knowledge. In both instances, they use face-to-face interaction and technology. Through face-to-face interaction, they help each other when they solve technical problems, during meetings and training sessions. Segal (2004, p. 4) is of the view that sharing knowledge by formal training, through a network or community and via knowledge tools is effective. The conclusion drawn is that for developers and organisations knowledge sharing practices are similar.

The Internet, email, and local area networks (LAN) technologies are tools widely used by developers and whole organisations for knowledge sharing purposes (Boden, Nett & Wulf, 2010; Kuusinen *et al*. 2017; Aurum, Daneshgar, & Ward, 2008). The email is used to share knowledge with seniors and it is also used to share with colleagues (Kuusinen *et al.,* 2017). Menolli *et al*. (2015, p. 296) identify Internet/Web 2.0 technologies such as blogs and social

networks and Wikis as mainly used by software organisations for knowledge sharing purposes. Zahedi, Shahin, and Babar (2016, p. 999) classify knowledge sharing tools in software organisations into three: communication and coordination tools, organisational memory tools, and project management tools. Communication tools allow organisations to share knowledge using text, sound and video. Organisational memory tools allow the capture and storage of knowledge and project management tools are for tracking project progress.

This discussion reveals that software organisations in the South African context share knowledge just like their global counterparts. They all share knowledge internally (among colleagues) and externally (with stakeholders and experts) using mostly Internet technologies and face-to-face interaction. These activities are also adopted by developers.

6.3.6. Knowledge storage

"After the identification or creation of knowledge it is codified and stored in corporate databases or knowledge based systems for retrieval and application or reuse" (Samoilenko & Nahar, 2013, p. 1353). This study revealed that software organisations keep their knowledge in organisational databases, servers, portals and personal computers. It was discovered that knowledge generated within the organisation is mostly kept electronically. In the software engineering field, knowledge is stored for reuse purposes (Basili *et al*., 1994, 2007). On the other hand, the study found that because of the size of some of the organisations, knowledge is not usually stored in a central place but in disparate systems scattered around the organisation. This is contrary to suggestions by Basili *et al*. (1994, 2007) who suggest that knowledge should be stored in a central institutional repository. They call such a repository an experience factory because it holds the experiences of the organisation. According to Basili *et al*. (1994, 2007), this concept describes a certain type of software organisation that captures and uses knowledge gained from past projects (experience) and uses it in current and future software projects. The idea is supported by Garcia *et al*. (2011) who argue that

knowledge in a central repository is retrieved easily and supports learning. Jansma and Means (2012) also support a central storage system and state that the National Aeronautics and Space Administration Agency (NASA) uses a centralised knowledge repository for their development tasks. They state that the system captures and stores all the professional knowledge within the agency and that users tap into this knowledge for their development work. Samoilenko and Nahar (2013) state that in software organisations knowledge is stored on the Internet, in printed form such as in manuals, books, documents, journal articles and in organisational databases. This study found that technology is the most widely used form of knowledge storage. This is consistent with studies from other parts of the world. Tools that have been found to be effective for knowledge storage are Wikis, blogs and organisational databases (Scherp, Schwagereit & Ireson, n.d; Kamunya & Waweru, 2013).

The results also indicate that developers store their knowledge in two main storage devices: servers including knowledge databases, Wikis and personal computers, and their heads. Two developers indicated that their knowledge is stored in their heads. The tools used are the same as the organisational tools used for knowledge storage.

Apart from software development organisations, technology seems to be used quite often in organisations in general. Olivera (2000, p. 819) found that organisations use social networking technologies, knowledge intranets, electronic bulletin boards, and knowledge databases to store organisational knowledge. Kasvi, Vartiainen, and Hailikari (2003, p. 576) list technologies such as the Internet and email as used for knowledge storage. Coakes, (2006, p. 581) mentions technologies such as expert systems, neural networks, fuzzy logic and intelligent agents as tools used for knowledge capture. Dingsøyr and Smite (2014) found that organisations use knowledge repositories such as corporate databases, intranets to store organisational knowledge.

What can be deduced from this discussion is that technology is mainly used by developers and organisations for knowledge storage.

6.3.7. Knowledge organisation

Once the knowledge has been stored it has to be organised. Knowledge organisation is the cataloguing, indexing, filtering, and the codification of knowledge for easy retrieval (Awad & Ghaziri, 2004, p.24; Samoilenko & Nahar, 2013, p.1356). In this study, it was revealed that some organisations do organise their knowledge while others do not. Those that organise knowledge do so in organisational servers and personal computers in folders. The assumption is that those that do not organise it, store it in different locations in the organisations as indicated earlier. The same was found with developers. The literature indicates the importance of organising knowledge in software organisations, but falls short of stating exactly where and how that knowledge is organised except to say it is organised in organisational databases and then further suggests ways to organise it (Chow & Wong, 2011; Mat & Liu, 2011). The conclusion drawn is that developers and organisations organise their knowledge in folders.

6.3.8. Knowledge application

For knowledge to be useful, it has to be applied in organisational routines and processes. In this study, it was found that knowledge is used/applied in the software development process and running the software development business. Alavi and Leidner (2001, p. 122) are of the view that knowledge application rather than the knowledge itself is an important aspect of the knowledge-based theory of the firm. Becerra-Fernandez and Sabherwal (2010) state that knowledge application happens when available knowledge is applied in organisational routines and processes to support decision making. Samoilenko and Nahar (2013, p.1280) state that knowledge application is the actual use of the knowledge for the benefit of individuals, teams, and the whole organisation. In this study, it was found that all

organisations and developers apply their knowledge in their development and that the application of knowledge has benefited both developers and whole organisations. Managers and developers narrated how code snippets and shared knowledge is applied in the development process and in the organisation. This is consistent with the software development and knowledge management literature. Baisch and Liedtke (1998) explain how knowledge extracted from a knowledge management system is applied in a software development environment. Knowledge application in software development is mostly associated with learning, namely how organisations react to current situations with the help of experience (past internalised knowledge). We conclude that software organisations and developers use their knowledge in their processes and routines.

6.3.9. Knowledge management benefits

Software organisations are adopting knowledge management practices because of the perceived benefits that knowledge management bring. This is because the software development process is a knowledge intensive task. Studies have long discussed the potential and actual benefits of applying knowledge management principles in software engineering. Lindvall, Rus, and Sinha (2002) are of the view that knowledge management supports the know-how, know-where, know-why, and the know-what of the organisation. They further state that knowledge management can help decrease delivery time, lower costs and increase quality.

This study found that knowledge management has benefited developers and organisations by saving time, retaining their intellectual property, increased effectiveness and efficiency and knowledge reuse. These findings are consistent with the literature. Sharma, Singh and Goyal (2012, p. 27) state that the major benefits of knowledge management in software organisations are saving time and software reuse. Candida, Natali, and Fablo (n.d.) state that software reuse prevents past failures from happening again. Other benefits they found were

that knowledge management enables the development of cheaper products, produces better product quality, and reduces the activity budget. Kumarawadu (2008, p. 263) concurs and states that knowledge management activities "improve organisational competitive advantage in terms of cost efficiency, excellence in quality, meeting timelines and innovative products." Sholla and Nazari (2011, p. 61) are of the opinion that knowledge management tools can benefit software organisations by lowering development costs. Chandani *et al*. (2007) and Bjornson and Dingsoyr (2008) are of the same view and state that knowledge management has improved the execution and coordination functions in the organisation. Tiwana (2004, p. 899) has found that knowledge management practices increase software development effectiveness, reduce defect density, lower warranty defects and increase software development efficiency. Silverman (2006) found that knowledge management practices improved the software development process.

Sharma, Singh and Goyal (2012) state that developers save a lot of time because of knowledge management. Rezende and Alves (n.d.) state that knowledge management has enabled developers to be effective by improving the skills and knowledge of the workers, thus improving worker efficiency and productivity, thus increasing the organisations' adaptation of products or services to client requirements

Other studies adopt other software development perspectives but also report on the benefits of knowledge management in those areas. For example, Chang (2013) is of the view that knowledge management has helped mitigate risk in software development. The same sentiments are shared by Lindvall, Rus, and Sinha (2002) who state that knowledge management can mitigate risk. Risk is prevented by preventing people repeating mistakes, preventing the unavailability of knowledgeable people in the organisation and preventing loss of knowledge owing to attrition. Abdou and Kamthan (2014) claim that knowledge management is essential in software testing because it allows the preservation of test

knowledge, thus preventing the "reinvention of the wheel". Knowledge management practices have also helped software development teams share knowledge, thus improving the software development process. This is evident in distributed teams in offshore organisations or teams in the open-source development community (Pinjani & Palvia, 2013). Knowledge management has also fostered team learning (Boden, Nett & Wulf, 2010; Menolli, Malucelli & Reinehr, 2011).

The findings of the study are consistent with what has been reported in the literature, namely it has confirmed that knowledge management has benefits for developers and software organisations at large. This is the case in South Africa.

6.3.10. Post-mortem reviews for organisational learning

Software organisations that apply knowledge management practices are called learning software organisations. This is because knowledge management is associated with learning, which in turn is associated with improved software development practices (King, 2009). Organisational learning aims to facilitate individual and group learning to advance the learning software organisation concept (Bennet & Bennet, 2003). One main learning activity that has been widely adopted by software development organisations is post-mortem reviews. According to Dingsøyr (2005, p. 293) conducting post-mortems is a method of organisational learning which can be organised for projects either when they end a phase or are terminated. They are used to reflect what happened in the past in order to improve future practice. Birk *et al.* (2002, p. 45) state that post-mortem reviews are an excellent knowledge management technique that can improve software development activities if properly applied. Desouza, Dingsøyr and Awazu (2005, p. 203) state that a post-mortem can be used to capture tacit experiences in a project. They further state that conducting a post-mortem is salient to identifying what is learned, what major issues were encountered, and what could be done to improve processes in the future. Anquetil *et al.* (2007, p. 528) confirm the importance of

post-mortem reviews in software development and concluded that they are capable of uncovering lessons from past projects, and that application and domain knowledge are obtained from such analysis. Baaz *et al*. (2010, p. 72) also confirm the importance of post-mortem reviews and state that they support knowledge codification, increase knowledge sharing within and across projects, improve worker participation and improve learning. Desouza, Dingsøyr and Awazu (2005, p. 205) give details of how individuals, teams and the whole organisation can benefit from post-mortem reviews. They state that at individual level, a post-mortem allows individuals to get feedback on their performance in the whole project. At team level, they enable teams to discuss and obtain feedback about the coordination and collaboration in the project. At organisational level, the review allows the organisation to capture the knowledge and make it available to the whole organisation. They conclude that if properly implemented they can help organisations implement effective and efficient software project management. In their post-mortem analysis, Ahonen and Savolainen (2010) identified key phases of knowledge loss in projects and suggested ways to improve the process through post-mortem reviews. Birk, Dingsøyr, and Stålhane (2002) caution that projects must never be closed before a post-mortem is conducted.

The literature clearly indicates that post-mortem reviews are a knowledge management activity that enables knowledge capture, codification, storage, sharing and facilitates learning. In fact, they are an integral part of the software development process. In this study, it was found that the majority of the organisations do not conduct post-mortem reviews. Even the few that indicated that they do, just conduct lightweight post-mortems during meetings. This is worrying because the literature clearly gives evidence that they play an important part in the knowledge management process and most importantly in the software development process. The study did not seek to find the reasons for not conducting post-mortem reviews, but managers hinted that time is the main barrier to conducting them. Baaz *et al*. (2010, p. 77)

also identified time as one of the barriers to conducting post-mortem reviews. They are supported by Bjørnson, Wang, and Arisholm (2009, p. 150) who state that "in the busy workday of a software project, there is rarely time for such reviews". Glass (2002) has similar sentiments and states that the software engineering field should be concerned with working smarter and not faster, but there is no time to think about smarter. Soon after a project is finished, developers are rushed into new projects without reflecting on the past project. Glass (2002) also states that sometimes organisations have no idea what a post-mortem must contain, which makes it difficult even to start it. Keegan and Turner (2001) also state that time projects face pressures in many organisations, which prevents them from learning. Apart from time barriers, Baaz *et al*. (2010) also found the following barriers: perceived high costs of information retrieval, getting lost in the current business, lack of commitment and incentives, no culture for inter-project learning, lack of mechanism to encourage exploitation, bad planning, impracticality to include all stakeholders in reviews, blindness towards one's own work, and situated knowledge.

Judging from the literature and the results of this study, it is clear that South African software SMMEs are bound to repeat past mistakes because they do not conduct post-mortem reviews, an activity that is so essential to the learning of the organisation. This means that each time they finish a project; they have no idea how the project has succeeded or failed. This denies the developers and the organisation valuable lessons that are learned. The conclusion therefore is that the high software failure rates reported in the study are caused by the fact that the organisations do not learn from their past mistakes, thus repeating them every time they develop software.

6.3.11. Knowledge management challenges

Though knowledge management has many advantages, it has its challenges too. In this study, it was found that software organisations face the following knowledge management challenges: a lack of formal knowledge management procedures, the fact that protecting the knowledge or IP is difficult, knowledge storage costs are high, organisations face never-ending needs for information, there is a lack of time to fully adopt knowledge management practices, difficulty in finding and learning from information, and ever-changing knowledge needs. In the software engineering field, little has been reported about the challenges faced by organisations. The literature tends to focus mainly on the benefits of knowledge management and ignores the challenges. General literature presents a few challenges. Gupta, Iyer and Aronson (2000) are of the opinion that organisational knowledge sharing culture and an emphasis on technology instead of a balance between it (technology) and people are barriers to the knowledge management course.

Soakell-Ho and Myers (2012, p. 212) state that organisational structure and culture, funding, and competition among organisational branches are known barriers to knowledge management. Kalkan (2008) mentions that dealing with tacit knowledge, conceptualising a working definition of knowledge management, utilising information technology, adapting to organisational culture, and coping with fierce competition are major knowledge management challenges. Kalkan states that it is important to define knowledge management to avoid duplication of effort, for example, knowledge management versus data and information management. A clear distinction will help organisations do the right thing. They further state that tacit knowledge is difficult to articulate, thus making it difficult to manage. They suggest that top management should support its management together with the use of IT. Smith and Lumba (2008, p. 167) have found that “managerial and internal controls, knowledge creation and sharing incentives, environmental influences relating to an

organisation's culture, resource influences, and the needs of partner organisations are major knowledge management challenges. Fontaine and Lesser (2002, p. 1) found that knowledge management challenges include failure to align knowledge management efforts with the organisation's strategic objectives. Vaezi (2005) cites time as the major barrier to the adoption of knowledge management. Vaezi (2005, p. 536) states that workers are willing to adopt knowledge management but are hindered by time pressure. This study concludes that as although there are knowledge management benefits, there are challenges too. Although little has been reported in the software engineering field about the challenges faced, we can compare them with other organisations. The context is slightly different, but it is not ignored completely. Knowledge management challenges in software engineering need further investigation. Therefore, the conclusion is that there are knowledge management challenges in organisations in general and challenges are also found in software development organisations.

6.4. Other relevant observations from data analysis

A further analysis and discussions of the results reveal that knowledge acquisition, creation and sharing processes are almost similar. For example, the Internet can be used for all three processes. Another example is a training session is used for knowledge acquisition, creation, and sharing. In such situations, it depends on the task of the individual or organisation. For example, when a developer is stuck in their tasks, they decide to go to the Internet to look for information. The Internet becomes a knowledge acquisition medium. When the developer reads and internalises the information from the Internet, that becomes knowledge creation. When developers meet in online forums and share their knowledge and experiences, that becomes a knowledge sharing environment. Segal (2004) concurs that these tasks are similar.

A further analysis of the results and comparison with the literature reveals that individual software developers' knowledge management practices are not different from the organisation wide practices. In the current study, for example, developers have the same knowledge needs as their organisations and use the same resources for knowledge acquisition, creation and sharing. This means that in the software organisations, knowledge management practices are practised by individual developers and get adopted by the whole organisation as indicated by Argyris and Schon (1978, 1996), Senge (1994) and Nonaka and Takeuchi (1995). The assumption is that the organisation facilitates the knowledge management process by providing the necessary infrastructure (especially technology) and a conducive environment to the employees.

It was also discovered that technology is widely used for knowledge management purposes in South African organisations, especially for knowledge sharing, storage, creation and acquisition. Internet technologies, especially Web 2.0 technologies (Wikis, blogs and social networks) and database systems are widely used in the organisations. This is consistent with studies by Dingsøyr and Smite (2014) who found that organisations use knowledge repositories such as corporate databases, intranets to store organisational knowledge, Wikis for knowledge mapping purposes and task boards to share and disseminate knowledge. Kamunya and Waweru (2013, p. 4) state that content management systems, knowledge sharing tools and knowledge search and retrieval tools are essential tools in the software organisation. Portillo-Rodríguez *et al.* (2012, p. 671) found that organisations use Wikis, document management systems and blogs for knowledge management purposes. Menolli *et al*. (2015. p. 296) say that Wikis, collaborative writing tools, social networks, repositories of shared documents, and property are used in software development environments. Most of the tools found in the literature are used by South African software development SMMEs.

6.5. Summary

This chapter provided further analysis, interpretations and discussions of the findings of the study according to the objectives of the study. The literature, the theoretical and philosophical underpinnings of the study were used to inform the discussions. The discussions revealed that software development failures are prevalent in South African software developing SMMEs. Projects are mostly cancelled or abandoned, and this is caused by ten factors: bureaucracy in IT departments, compatibility issues, complacency of developers, involvement of wrong people in the planning stages of projects, lack of detailed documentation, lack of resources, lack of user commitment, miscommunication of requirements, unrealistic outcomes, and work overload. This is not only a South African problem, but a global one as indicated in the literature.

The chapter also revealed that organisations experience technical and business knowledge needs which they satisfy by looking for knowledge from internal and external sources. The Internet is the main source of knowledge for organisations and for developers. It was also revealed that organisations and developers share knowledge with their colleagues internally and externally with the help of technology especially emails and the Internet. In the organisations, knowledge is created by reading and internalising the knowledge. It is also created by the socialisation process when tacit knowledge is received from colleagues within and outside the organisation. This happens at organisational and individual level. The available knowledge is stored in organisational servers, personal computers, Wikis, and portals, and it is organised in simple folders. It was satisfying to learn that organisations and developers apply the knowledge they have obtained in their tasks and that has come with benefits. Organisations and developers reported increased effectiveness and efficiency, improved product quality and time saved. These results were consistent with findings in the global literature. Unfortunately, two very important knowledge management practices were

not fully adopted by organisations, namely formal training and conducting post-mortem reviews. Time is believed to be the major barrier to the adoption of these activities. Technology plays an important role in the management of knowledge in the organisations. Four conclusions are drawn from these discussions; 1) South Africa software developing SMMEs experience software project failures, 2) developers and organisations have adopted knowledge management practices and the practices are consistent with the theoretical framework developed for the study, 3) developers and organisations have benefited from these practices, and 4) knowledge management comes with challenges.

CHAPTER SEVEN
SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

7. Introduction

This study sought to understand the nature of software development failures in software development SMMEs in South Africa with a view of exploring how KM could be used to alleviate the failures. The study also sought to investigate the benefits and challenges that accompany knowledge management. Specifically, the study sought to answer the following questions:

1. What is the form and source of software development failures experienced by software development SMMEs?
2. What knowledge management practices are adopted by individual software developers and software development SMMEs in South Africa?
3. What benefits has knowledge management brought to software developers and organisations?
4. What knowledge management challenges do software development SMMEs face?

7.1. Summary of chapters

Chapter one presents details of the research background and identifies the research problem and the questions that this study sought to answer. The chapter also provides the summary of the whole study.

Chapter two presents a conceptual and theoretical framework that was developed from two fields of study: knowledge management and software engineering. The aim of the conceptual and theoretical framework was to define concepts relevant to this study and use them as a guide for the study. It explains what knowledge management practices are in the context of this study. Knowledge management practices were built from 15 existing knowledge management frameworks. They are knowledge acquisition, creation, storage, sharing, organisations, and application. Software engineering concepts relevant to this study were also

defined. They are software development failures, types and causes. A detailed discussion of the conceptual and theoretical framework is presented in Chapter two.

Literature relevant to the study was reviewed and presented in Chapter three. The focus of the literature review was software development and software development failures. The review revealed that software development failures are prevalent in software organisations in South Africa and globally and they are caused by many factors including failure to learn from the past and lack of skills. The literature survey also discussed knowledge management practices in software organisations. Four areas of knowledge management research were discovered in the software engineering field. They are knowledge management in software organisations, in the software development process, in testing, and development methodologies, in mitigating risk and the role of knowledge management in fostering learning in software development organisations. In all these areas of research, the literature reveals that software organisations adopt different knowledge management practices for different reasons. For example, some organisations adopt knowledge management to identify knowledge gaps, others to create knowledge repositories and others for learning purposes. In the South African context, such studies are lacking. Very little is known about knowledge practices, benefits and challenges in the South African context. There is also very little literature on individual software developers' knowledge management practices. Only one South African study investigated developers; its focus was knowledge sharing among developers. This study addressed these literature gaps.

Idealism was adopted as an ontological philosophy and the interpretive paradigm was adopted as an epistemological philosophy underpinning the study. Interpretivism enabled the researcher to interpret the responses of the respondents and to draw conclusions. This was a qualitative multi-case study conducted in the province of KwaZulu-Natal, South Africa targeting software development SMMEs. SMMES were targeted because they are the major software developers in South Africa. Twelve SMMEs participated and data were collected through interviews with 12 software development project managers and eight software developers. They were targeted because they are the people mainly responsible for software development. Content analysis was used to analyse and interpret the data, from which conclusions were drawn. Detailed discussions of the methodology are found in Chapter four.

The results of the study are presented in Chapter five. The main findings were that software development companies face software development failures and that they have adopted knowledge management. Knowledge management has benefited the organisations but there are challenges. Software developers have adopted knowledge management too, and are starting to see the benefits.

The discussions of the results (Chapter six) discovered that the results of this study are consistent with many findings in the literature, but there are also findings unique to the South African context. Details are given in the following sub-sections.

7.2. Summary of findings

This section summarises the findings of the study. The summary is presented according to the research questions.

7.2.1. Are software development SMMEs experiencing software development failures?

This question was aimed at determining if software development failures occur in South African software development SMMEs and to further investigate if knowledge management plays a role in reducing or eliminating the failures. The study found that:

- Organisations experience software development failures. All but two organisations indicated that they have experienced software development failures in their lifetime.
- The failures are either complete or partial (challenged) failures. The majority are complete failures.
- There are ten causes of software development failures in South Africa. These are:
    - Bureaucracy in IT departments
    - Compatibility issues
    - Complacency of developers
    - Involvement of wrong people in the planning stages of projects
    - Lack of detailed documentation
    - Lack of resources
    - Lack of user commitment
    - Miscommunication of requirements
    - Unrealistic outcomes
    - Work overload.

The discussions of the results in Chapter six identified similar results in studies from other developing and developed countries (Egbokhare, 2014; Ebad, 2016; Vithana, Fernando, & Kapurubandara, 2015). However, two causes of failure were found to be unique to South Africa. These are bureaucracy in IT departments and the involvement of wrong people in the planning stages of projects.

7.2.2. What knowledge management practices are adopted by software SMMEs in South Africa?

The prevalence of software development failures has forced software development organisations to adopt knowledge management (Lindval, Rus, & Sinha, 2002; Sholla & Nazari, 2011). This is because of the knowledge intensive nature of software development and that interventions by SPI frameworks and agile methods do not seem to be successful in reducing or eliminating the failures (Al Tarawneh, Abdullah, & Ali, 2009).

This question was divided into two parts: knowledge needs and knowledge management practices. First, the knowledge needs of the organisations had to be determined before the actual practices were investigated. This question was asked because the assumption was that determining the knowledge needs would justify the adoption of knowledge management practices in the organisations.

The study found that software development organisations have two types of knowledge needs: technical and business knowledge needs. Technical knowledge needs are related to programming or coding knowledge and skills or other technical skills necessary in the actual development of software. Business knowledge is knowledge about the industry within which the organisations are developing the software product and knowledge pertaining to the general software development industry. This is competitors' and stakeholders' knowledge. The literature review (Chapter three) presents similar results in a global context.

The second part of this question investigated the actual knowledge management practices adopted by the organisations. A knowledge management conceptual and theoretical framework (Chapter two) was developed to answer this question. The results indicate that software companies have adopted the six knowledge management practices. The activities are:

- *Knowledge acquisition* – knowledge is acquired from internal and external sources. Internally, it is acquired from organisational repositories (documents), and from colleagues. Externally, it is acquired from industry networks,

customers and the Internet. The Internet was found to be the main source of external knowledge.

- *Knowledge creation* – knowledge is created by internalising information acquired from internal and external sources. When the actors acquire and internalise the information, they create their own knowledge which they apply in their tasks. Formal training, which is a widely used method of knowledge creation, is hardly used by the organisations. The study assumes that the lack of resources is the main cause. In many developing countries SMMEs lack the financial and material resources to offer formal training. Robertson (2003, p. 461) states that SMMEs "often lack the knowledge and resources to engage in training programs".
- *Knowledge storage* – organisations store their knowledge in central organisational repositories. IT plays a huge role in knowledge storage. Organisational servers are mostly used for knowledge storage. In some organisations, knowledge is not stored in central places, but it is scattered in employees' personal computers.
- *Knowledge organisation* – knowledge is organised in personal computers or in organisational servers in folders relevant to each piece of information.
- *Knowledge sharing* – knowledge is shared internally and externally. Internally, it is shared between colleagues, and externally with peers in the same network and with customers.
- *Knowledge application* – the knowledge that organisations have is applied in their development tasks. For example, stored code snippets are copied and pasted in new projects.

The same results were found in developers' knowledge management practices. This is because the same conceptual and theoretical framework was used to investigate their knowledge management practices. It is also assumed that this is because they use organisational infrastructure for their knowledge management activities. Aurum, Daneshgar, and Ward (2008) also found that the knowledge management activities of developers are the same as those of the organisations.

Another knowledge management activity that is widely adopted in software development organisations is post-mortem reviews (Lehtinen *et al*., 2014). The study found that most South African software organisations do not formally conduct post-mortem reviews. Others do not do them at all. The main reason given by organisations was that they do not have the time to conduct them.

7.2.3. What benefits has knowledge management brought to the SMMEs?

Organisations were asked to state if there are any benefits that they have gained by adopting knowledge management practices. This question was asked because globally software organisation are starting to reap the rewards of knowledge management (Bjornson & Dingsoyr, 2008; Kristjansson, Helms, & Brinkkemper, 2014). The study sought to establish if the same happens in South Africa and to reveal the actual benefits in the South African context. The study revealed that software organisations are benefiting from knowledge management practices. Ever since they adopted knowledge management they have improved their efficiency and effectiveness, saved time, retained their knowledge resources and are able to reuse their knowledge. These are the same benefits that are enjoyed by many other organisations worldwide, as indicated by Chandani, Neeraja and Sreedevi (2007) and Kristjansson, Helms and Brinkkemper (2014). Software developers reap the same benefits as their respective organisations.

7.2.4. What knowledge management challenges do software SMMEs face?

The software engineering literature says very little about knowledge management challenges experienced by software organisation. This study sought to investigate this issue to get an understanding of the knowledge management challenges that organisations might face. The study found the following challenges:

- Lack of formal knowledge management procedures in organisations
- The difficulty of protecting the knowledge
- Expensive storage costs
- The never-ending need for information
- Lack of time to adopt knowledge management practices fully
- Difficulty in finding and learning from information
- The ever-changing knowledge needs.

7.3. Conclusions of the study

The conclusions focus on salient issues, the gaps, the implications, contributions and originality and the novelty of the study. The following issues emerge from this study:

7.3.1. Software development failures

The study concluded that software development failures are prevalent in South African software SMMEs, just as it was found by De Wet and Visser (2013) and later by the ITWeb Reports (2011, 2012, 2013). The study also confirms results of global studies, specifically the influential Standish Group Report (2015). It is further concluded that the failures are caused by ten factors as indicted in Chapter five. When the causes are compared to global studies and studies in developing countries, the study concluded that two causes are unique to South African organisations. They are bureaucracy in IT departments and the involvement of wrong people in the planning stages of projects. The literature (Chapter three) indicates that this problem has been persisting for quite some time in developing and developed countries. Memories of the failed software project implementation by the Gijima Group at the South Africa Department of Home Affairs in the year 2007 are still fresh in the minds of many South Africans. As this problem continues to persist, software development companies and their clients continue to lose money and time developing software that is not going to be used. This causes disruptions to the operations of businesses. As stated by Lindval, Rus and Sinha (2002) and Sholla and Nazari (2011), the full adoption of knowledge management practices could be the solution to this problem. This is because the same causes are reported every year and there are strategies in place attempting to find a solution to this issue. All that is in place seems to be failing. The question that this study asks is: why do software organisations fail to address a well-known problem? The study assumes that failure to learn from past mistakes is the main reason and concludes that the problems are going to be addressed only by knowledge management as suggested by Cerpa and Verner (2009) and Lyytinen and Robey (1999).

7.3.2. Organisational knowledge needs

Project managers and developers indicated that their organisations do not have all the expertise needed to run a software development organisation. This creates knowledge gaps in the organisations, which they fill by acquiring knowledge from many sources. As indicated in Chapters five and six, the study concluded that software development organisations require technical and business knowledge. The same types of knowledge are also required by their developers. Mishra and Mahanty (2015) found that software projects require high levels of

business knowledge. This study recommends the blending of technical and business knowledge in the software development industry for maximum benefits. These results are consistent with the literature in Chapter three and therefore the conclusion drawn is that South African organisations' knowledge needs are not unique compared to global knowledge needs.

7.3.3. Knowledge management practices

There are six knowledge management practices that organisations and software developers have adopted: knowledge acquisition, creation, storage, sharing, organisation and application. This is in line with the conceptual and theoretical framework presented in Chapter two. These findings are unique to the South African context. This is because the framework that was used to investigate this phenomenon has never been used in any study before. On the other hand, there are similarities found in other studies in the literature when the practices are detached from the framework of this study and investigated individually or used in other, different frameworks. Examples are Basri and O'Connor (2011), Heavin and Adam (2012) and others. Detailed examples are given in Chapter three. In summary, the study concluded that South Africa software organisations have adopted knowledge management practices and that there are six knowledge management activities adopted in the South African context. The study further concluded that knowledge acquisition, creation and sharing are similar in that they use the same sources of knowledge. For example, the Internet is used for all three activities. When looking at each knowledge management activity, the study concluded that knowledge is acquired from three main sources; the Internet – which is the main source – colleagues within and outside the organisation and organisational documents. Another conclusion drawn is that in many organisations, knowledge is stored in central databases and in others it is scattered throughout the organisation. Once the knowledge is stored, is it organised in folders and is applied in the day-to-day operations of the organisation, especially development work as stated by Aurum, Daneshgar, and Ward (2008) and Heavin and Adam (2012). Knowledge is created by socialisation and internalisation processes as defined by Nonaka and Takeuchi (1995), and is shared within and outside the organisations using mainly technology.

The study also concluded that two crucial knowledge management practices are not adopted in South African software development organisations. The activities are formal training and post-mortem reviews. This is contrary to suggestions given in the literature. Baaz *et al*. (2010) highly recommend post-mortem reviews for software organisations and Sharma,

Singh and Goyal (2012) recommend formal training for software developers. Time was given as a barrier to conducting post-mortem reviews.

7.3.4. Knowledge management benefits

The software organisations have benefited from knowledge management practices. Knowledge management has enabled them to save time, they have become more efficient and effective and it has enabled them to retain their intellectual property as indicated by Chandani, Neeraja and Sreedevi (2007) and Kristjansson, Helms and Brinkkemper (2014). This is consistent with the literature as presented in Chapter three. The study did not find benefits unique to the South African context. What is not clear is the role(s) that knowledge management plays in eliminating or reducing software development failures. The ten causes of software development failure found in this study are not directly related to knowledge management. It is therefore concluded that although organisations are reaping knowledge management benefits, the role of knowledge management in addressing failures is unknown.

7.3.5. Knowledge management challenges

The study concluded that knowledge management has brought benefits to organisations, but there are also challenges in managing knowledge. These findings relating to challenges are unique to South Africa. The literature is silent about knowledge management challenges specifically in software development organisations. It is the general knowledge management literature that reports challenges in organisations in general, and most of the studies do not mention the findings of this study. They mention organisational culture, lack of top management support, dealing with the nature of knowledge and other unrelated challenges (Smith & Lumba, 2008; Kalkan, 2008; Soakell-Ho & Myers, 2012). It is only the time challenge that appears in this study and existing literature. The study concludes that South African software organisations face the following challenges: lack of formal knowledge management procedures, protecting the knowledge, storage costs, the never-ending need for information, lack of time to fully adopt knowledge management practices, difficulty in finding and learning from information, and ever-changing knowledge needs.

7.3.6. Knowledge management practices of software developers

The study concluded that individual software developers have adopted the same knowledge management activities as their organisations. The assumption is that this is because they use organisational resources for their individual knowledge management activities. Knowledge management activities are adopted at individual and organisational levels. Argyris and Schon (1978) state that for organisations to learn, it is the individuals that have to learn first, then

teams which then leads to the whole organisation learning. In this study's context, organisations provide the infrastructure for knowledge management and the developers use it for their knowledge management activities.

7.3.7. Overall study conclusions

There are four main conclusions from this study. Firstly, the study concluded that South African SMMEs experience software development failures and that there are ten causes of the failures, of which two are unique to South Africa. South African software organisations face mainly absolute failures, which is quite strange because the literature indicates that most software projects are challenged (ITWeb Reports, 2011, 2012, 2013; the Standish Group Report, 2015).

Secondly, the study concluded that software development organisations and software developers have adopted knowledge management, but informally. We conclude that it is informal because organisations did not mention that they have knowledge management policies and specialists hired to drive the knowledge management initiatives. The informal nature of the adoption is further confirmed by the challenges that the organisations gave. One of the challenges is the non-adoption of formal knowledge management practices (Chapters five and six). This confirms Aurum, Daneshgar, and Ward's (2008) study which found that software SMMEs' knowledge management practices are informal. Drawing from Argyris and Schon (1978, 1996) the study also concluded that the adoption of knowledge management practices by software developers influences the adoption of the practices at organisational level. Argyris and Schon (1978, 1996) are of the view that learning by the individual leads to the learning of the whole organisation. The role of the organisation is to provide the infrastructure to support the learning activities. In this study, the organisations provide the knowledge management infrastructure that developers use.

Two vital knowledge management practices are ignored by organisations. They are formal training and post-mortem reviews. Birk, Dingsøyr and Stålhane (2002) caution software organisations never to leave a project without a post-mortem review, because they believe that it is an excellent way to capture experiences and improvement suggestions that is used in future projects. Dingsøyr (2005) believes that they are a simple and practical way to facilitate organisational learning. Post-mortem reviews can help the organisations identify lessons learned and best practices. The study concludes that one of the reasons project failures continue to occur is the organisations' inability to learn from their past mistakes. This

is because software organisations do not conduct post-mortem reviews, which are the most effective way of organisational learning. Lyytinen and Robey (1999) concur with this view.

Thirdly, the study concluded that software organisations have benefited from knowledge management practices by saving time, improved efficiency and effectiveness and keeping their intellectual property. This view is shared by Chandani, Neeraja and Sreedevi (2007). The unfortunate part is that the benefits do not seem to address the biggest problem facing software organisations, which is project failures. The study did not determine why is this happening.

Fourthly, the study concluded that although organisations are enjoying the benefits of knowledge management, there are challenges too. The challenges are associated mainly with costs associated with keeping the knowledge, and the ever-changing knowledge needs.

7.4. Contributions to knowledge

A doctoral study is expected to contribute new knowledge to the field(s) of study. It could be theoretical, methodological and practical. Corley and Gioia (2011, p.15) are of the view that theoretical contribution is the ability to provide original insights into a phenomenon for a useful purpose. They define original as increasing the understanding of a phenomenon in a scientifically or practically useful way. Methodologically, the study has not made a meaningful contribution to knowledge. This is because many studies which have investigated similar phenomena have used the same methodology as the current study as indicated in Chapter four. The study has made theoretical and practical contributions. They are discussed in detail in the following subsection.

7.4.1. Theoretical contribution

A conceptual and theoretical framework was developed for the purposes of this study. The framework is derived from two fields of study: knowledge management and software engineering. The framework was used to interpret knowledge management practices and software development failures in software development SMMEs. This is the first study to have used this framework. This is the main theoretical contribution. The study has also contributed to the existing body of literature in developing countries especially South Africa about knowledge management activities in the software engineering field.

7.4.2. Contribution to practice

In practice, the study has made two contributions. Firstly, the conceptual and theoretical framework developed will be used by software organisations to inform knowledge management activities. Software organisations will identify important areas to focus on to improve their activities. The study has also discovered knowledge management practices in software development organisations and recommended practical solutions to improve them for the maximum benefit. Practical solutions are that organisations have to hire knowledge management specialists to drive their knowledge management activities. It also recommends the formal adoption of post-mortem reviews to identify bad practices so as to prevent them in future and to identify good practices. These practices will improve organisational routines and processes, thus improving their software development processes and software quality. Secondly, the study has discovered the main causes of software development failure and knowledge management challenges. The software development industry is expected to use the findings to produce solutions to the software development failure problems and also solutions to the knowledge management challenges.

7.5. Recommendations

This study makes the following recommendations to practice:

1. It is recommended that software development SMMEs formally adopt knowledge management. This will be done by employing knowledge management specialists to drive knowledge management initiatives. A specialist will advise the organisation about the best knowledge management practices to be implemented and will act as a facilitator and custodian of the knowledge management initiatives, helping in the development of knowledge management policies. A full-time specialist will have the time, and with the support of management, have the necessary resources to support knowledge management initiatives.
2. The study found that some organisations do not store their knowledge in a central place. It is therefore recommended that those who do not yet do so should store their knowledge in central repositories for easy access and retrieval. Basili *et al*. (2007) and García *et al.* (2011) suggest central storage places for knowledge in software development organisations. They are of the view that central storage places enable easy storage and retrieval of knowledge.
3. Organisations should adopt formal training methods to update employees' knowledge and skills. The software development environment changes from time to time, thus

requiring regular specialist training. Aurum, Daneshgar, Ward (2008, p. 511) suggest that there is a need to formalise knowledge sharing in software organisations while ad-hoc training methods are also retained. Formal training will expose individual developers to the latest knowledge and skills of the industry.

4. The importance of post-mortem reviews in software development organisations has been reported in many studies (Birk, Dingsøyr & Stålhane, 2002; Dingsøyr, 2005). It is recommended that South African SMMEs, as a matter of urgency, conduct post-mortem reviews at the end of every project or during specific phases of projects. This is one of the tasks given to the knowledge management specialists if developers and managers have no time to do it.

Apart from recommendations to software development organisations, the study has recommendations for further research. It recommends that:

1. Further research is conducted to investigate how software development failures could be eliminated or reduced, especially investigating the role that could be played by knowledge management. The literature (Silverman, 2006; Gendreau & Robillard, 2013) has indicated that knowledge management can play a role, but that role has yet to be clearly identified.
2. Quantitative studies are conducted to measure the actual time and costs saved by implementing knowledge management. Previous studies (Chandani, Neeraja & Sreedevi 2007; García *et al.*, 2011) and this study found that knowledge management offers benefits to software development SMMEs in terms of saving time, efficiency and effectiveness and knowledge reuse.

References

Abdou, T., & Kamthan, P. (2014). *A knowledge management approach for testing open source software systems*. Paper read at the IEEE 33rd International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA.

Abdullah, R., Eri, Z. D., & Talib, A. M. (2011). A model of knowledge management system in managing knowledge of software testing environment. *Paper read at the 5th Malaysian Conference in Software Engineering (MySEC)*, Johor Bahru, Malaysia.

Abor, J. & Quartey, P. (2010). Issues in SME development in Ghana and South Africa. *International Research Journal of Finance and Economics*, 39, 218-228.

Addo, T.B.A., & Jennex, M.E.C. (2005). Issues in knowledge management strategy**.** In M. Khasrow-pour, *Managing Modern Organisations with Information Technology* (pp. 536 - 559). London: IDEA Group Publishing.

Ahonen, J.J., & Savolainen, P. (2010). Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects. *The Journal of Systems and Software*, 83, 2175–2187.

Ahsan, S. & Shah, A. (2006). Data, information, knowledge, wisdom: A doubly linked chain? Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.5378&rep=rep1&type=pdf

Akhavan, P. & Dehghani, M. (2015). Knowledge acquisition techniques selection: A comparative study. *The IUP Journal of Knowledge Management*, 13(3), 17-30.

Al Tarawneh, M.Y., Abdullah, M.S., & and Ali, A.B.M. (n.d.). Software Process Improvement (SPI) In Small Software Firms. Retrieved from http:// http://www.zuj.edu.jo/conferences/ICIT09/PaperList/Papers/Software%20Engineering/594mejhem.pdf.

Alagarasamy, K., Justus, S., & Iyakutti, K. (2006). A theoretical perspective on knowledge based organizational learning. *Proceedings of the XIII Asia Pacific Software Engineering Conference*, Kanpur, India.

Alavi, M. & Leidner, D. (2001). Review: knowledge management and knowledge management systems: conceptual foundations and research issues. *MIS Quarterly*, 25(1), 107-136.

Alhawaria, S., Karadshehb, L., Taletc, A. N., & Mansoura, E. (2012). Knowledge-based risk management framework for information technology project. *International Journal of Information Management, 32*(1), 50–65.

Alhawaria, S., Karadshehb, L., Taletc, A. N., & Mansoura, E. (2012). Knowledge-based risk management framework for information technology project. *International Journal of Information Management, 32*(1), 50–65.

Aljuwaiber, A. (2016). Communities of practice as an initiative for knowledge sharing in business organisations: a literature review. *Journal of Knowledge Management*, 20(4), 731-748.

Allard, S. (2003). Knowledge Creation. In C.W. Holsapple (ed.), *Handbook on Knowledge Management Volume 1*(pp. 367-380). Heidelberg: Springer –Verlag.

Althoff, K-D., Bomarius, F., & Tautz, C. (2000). Knowledge management for building learning software organizations. *Information Systems Frontiers*, 2 (3/4), 349-367.

Alwan, M. (2015) What Is System Development Life Cycle?" *Airbrake,* retrieved from http://airbrake.io/blog/sdlc/what-is-system-development-life-cycle. Accessed 28 Jan. 2017

Amaratunga, D., Baldry, D., Sarshar, M., & Newton, R. (2002). Quantitative and qualitative research in the built environment: application of mixed research approach. *Work Study*, 51(1), 17-31.

Amayah, A.T. (2013). Determinants of knowledge sharing in a public sector organization. *Journal of Knowledge Management*, 17(3), 454-471.

Ambler, S.W. (2008). Agile software development at scale. In B. Meyer, J.R. Nawrocki, and B. Walter (Eds.), *CEE-SET 2007, LNCS 5082*, (pp. 1–12), Heidelberg, Springer.

Andrade, J., Ares, J., García, R., Rodríguez, S., & Suárez, S. (2006). A reference model of knowledge management in software engineering. *Engineering Letters*, 13(2).

Anquetil, N., de Oliveira, K.M., de Sousa, K.D., & Dias, M.G.B. (2007). Software maintenance seen as a knowledge management issue. *Information and Software Technology*, 49, 515–529.

Antunes, B., Seco, N., & Gomes, P. (2007). *SRS: A software reuse system based on the semantic web*. Paper presented at the 3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE) of the 4th European Semantic Web Conference (ESWC), June 2007.

April, K. & Izadi, F.A. (2004). *Knowledge management praxis*. Cape Town: Juta and Co. ltd.

Arent, J., & Nørbjerg, J. (2000). Software process improvement as organizational knowledge creation: A multiple case analysis. *Proceedings of the 33rd Hawaii International Conference on System Sciences, Wailea, Hawaii, January 2000* (pp. 11).

Argyris, C. & Schon, D. (1978) *Organizational learning: a theory of action perspective*, Reading: Addison-Wesley Publishing Company.

Argyris, C. (1977). Double loop learning in organisations. *Harvard Business Review*, 115-125.

Argyris, Chris & Schön, Donald A. (1996). *Organizational Learning II. Theory, Method and Practice*. Reading: Addison Wesley.
Arinloye *et al*. (2015). Taking profit from the growing use of mobile phone in Benin: A contingent valuation approach for market and quality information access. *Information Technology for Development,* 21(1), 44 – 66.

Armstrong, A., & Foley, P. (2003). Foundations for a learning organisation: organisation learning mechanisms. *The learning organisation*, 10(2), 74-82.

Attarzadeh, I. & Ow, S.H. (2008). Project management practices: Success versus failure. *Proceedings International Symposium On Information Technology*, Kuala Lumpur, Malaysia.

Au, Y.A., Carpenter, D., Chen, X., & Clark, J.G. (2009). Virtual organizational learning in open source software development projects. *Information & Management*, 46, 9–15.

Aurum, A., Daneshgar, F. & Ward, J. (2008). Investigating knowledge management practices in software development organisations – An Australian experience. *Information and Software Technology*, 50, 511–533.

Awad, M.A., & Ghaziri, H.M. (2004). *Knowledge management (2nd Ed.).* Upper Saddle River: Pearson Education.

Baaz, A., Holmberg, L., Nilsson, A., Olsson, H.H., & Sandberg, A.B. (2010). Appreciating lessons learned. *IEEE Software*, 72-79.

Baisch, E. & Liedtke, T. (1998). Automated knowledge acquisition and application for software development projects. *Proceedings. 13th IEEE International Conference on Automated Software Engineering*, Honolulu, Hawaii, USA.

Bapir, M.A. (2012). Is it possible for qualitative research to be properly valid and reliable? Retrieved from https://www.academia.edu/997438/Validity_and_Reliability_in_Qualitative_Research

Basili, V.R., Caldiera,G., & Rombach, D.H. (1994). The experience factory. *Encyclopedia of Software Engineering*, 2, 469-476. Wiley & Sons, Inc., 1994.

Baškarada, S. (2014). Qualitative Case Study Guidelines. *The Qualitative Report*, 19(24), 1-18.

Basri, S. & O'Connor, R.V. (2011). A study of knowledge management process practices in very small software companies. *American Journal of Economics and Business Administration*, 3(4), 636-644.

Becerra-Fernamdez, I., Gonzales, A., & Sabharwal, R. (2004). *Knowledge management: challenges, solutions, and technologies.* Upper Saddle River: Pearson Prentice Hall.

Becerra-Fernandez, I. & Sabherwal, R. (2010). *Knowledge management: systems and processes*. Armonk (N.Y.); London: M.E. Sharp.

Beecham, N.A.S. & Mistrík, I. (2010). Architectural knowledge management in global software development: a review. *Proceedings of the International Conference on Global Software Engineering*, Princeton, New Jersey, USA.

Begel, A. & Nagappan, N. (2007) Usage and perceptions of agile software development in an industrial context: An exploratory study. *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, Washington, DC, USA.

Bellini, E. & lo Storto, C. (2006). CMM implementation and organizational learning: Findings from a case study analysis. *Proceedings of the PICMET 2006,* (pp. 1256-1271), Istanbul, Turkey.

Benbasat, I., Goldstein, D.K., & Mead, M. (1987). The case research strategy in information systems. *MIS Quarterly,* 11(3), 369-386.

Bennet, A., & Bennet, D. (2003). The partnership between Organisational learning and knowledge management. Retrieved from http://w3.ualg.pt/~mzacaria/gic2011/cap23.pdf

Betz, S., Oberweis, A. & Stephan, R. (2014). Knowledge transfer in offshore outsourcing software development projects: An analysis of the challenges and solutions from German clients. *Expert Systems*, 31(3), 282–297.

Bhandar, M., Shan-Ling, P., & Tan, B.C.Y. (2005). Integrating knowledge across organisations: the role of social capital. In M. Khasrow-pour, *Managing Modern Organisations with Information Technology,* (pp. 484 -487). London: IDEA Group Publishing.

Bierly, P.E., Kessler, E.H., & Christensen, E.W. (2000). Organizational learning, knowledge and wisdom. *Journal of Organizational Change Management*, 13(6), 595 -618.

Bilal, Z.O. & Mqbali, N.S. (2015). Challenges and constrains faced by small and medium enterprises (SMEs), in Al Batinah governorate of Oman. *World Journal of Entrepreneurship, Management and Sustainable Developmen*t, 11(2), 120-130.

Birk, A., Dingsøyr, T., Lindstaedt, S.N., & Schneider, K. (2006). Learning software organisations and requirements engineering: first international workshop (LSO+RE 2006). *Journal of Universal Knowledge Management*, 1(2), 59-68.

Birk, A., Dingsøyr, T., Stålhane, T. (2002). Postmortem: never leave a project without It. *IEEE Software,* 19(3), 43-45.

Bjornson, F.O., & Dingsoyr, T. (2008). A study of a mentoring program for knowledge transfer in a small consultancy company. *Lecture Notes in Computer Science 3547(*pp. 245-256*)*, Heidelberg, Springer Verlag.

Bjørnson, F.O., Wang, A.F., & Arisholm, E. (2009). Improving the effectiveness of root cause analysis in post-mortem analysis: A controlled experiment. *Information and Software Technology,* 51, 150–161.

Blackler, F. (1995). Knowledge, knowledge work and organizations: an overview and interpretation. *Organization Studies,* 16(6), 1021-1046.

Blaikie, N. (2010). *Designing social research: the logic of anticipation* (2nd Ed.). Cambridge: Polity Press.

Boden, A., Nett, B., & Wulf, V. (2010). Operational and strategic learning in global software development. *IEEE Software*, 27(6), 58-65.
Boehm, B. (2002). Get ready for agile methods, with care. *IEEE*, 64-69.

Bogopa, M.E. (2010). The maturity of IT project management in South Africa. Retrieved from http://www.bogopa.co.za/Research%20For%20IT%20Project%20Management%20Maturity%20Model.pdf.

Bollou, F. & Ngwenyama, O. (2008) Are ICT investments paying off in Africa? An analysis of total factor productivity in six West African countries from 1995 to 2002. *Information Technology for Development*, 14(4), 294-307.

Boddie, J. (1987). The Project Postmortem. *Computerworld*, December.

Bostrom, P. R. & Heinen, J. S. (1977). MIS problems and failures: a socio-technical perspective part I: The causes, *MIS Quarterly*, 1(1), 17-32.

Bowen, S. & Maurer, F. (2002). Process support and knowledge management for virtual teams doing agile software development. *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02),* Oxford, England.

Braganza, A. (2004). Rethinking the data – information – knowledge hierarchy: towards a case-based model. *International Journal of Information Management*, 24, 347 – 356.

Brink, H.I.L. (1993). Validity and reliability in qualitative research. *Curationis,* 16(2), 35-38.

Brooke, C. (2002). Editorial: critical research in information systems. *Journal of Information Technology,* (1)17, 45-47.

Bukowitz, W.R., & Williams, R.L. (1999). *The knowledge management fieldbook*. London: Prentice Hall.

Bupa, S.H. (2005). Why do so many major IT projects fail? *Computer Fraud & Security,* 15-17.

Burrell, G., & Morgan, G. (1979). *Sociological paradigms and organisational analysis.* London: Heinemann.

Business Tech, (2017). Top 5 challenges for small businesses in South Africa. Retrieved from https://businesstech.co.za/news/technology/151449/top-5-challenges-for-small-businesses-in-south-africa/

Cao, L. & Ramesh, B. (2007). Agile software development: Ad hoc practices or sound principles?.*IT Pro*, 41-47.

Cecez-Kecmanovic, D., Kautz, K., & Abrahall, R. (2014). Reframing success and failure of information systems: a performative perspective. *MIS Quarterly*, 38(2), 561-588.

Cerpa, N., & Verner, J.M. (2009). Why did your project fail? *Communications of the ACM*, 52(12), 130-134.

Cerpa, N., & Verner, J.M. (2009). Why did your project fail? *Communications of the ACM*, 52(12), 130-134.

Chaffey, D., & Wood, S. (2005). *Business information management: improving performance using information systems*. London: Pearson Education Limited.

Chan, J. & Husted, K. (2010). Dual allegiance and knowledge sharing in open source software firms. *Creativity and Innovation Management*, 19(3), 314-326.

Chandani, A., Neeraja, B., & Ms. Sreedevi (2007). Knowledge management: an overview and its impact on software industry. *Proceedings of the IET-UK International Conference on Information and Communication Technology in Electrical Sciences* (ICTES 2007), Chennai, Tamil Nadu, India.

Chang, C-P. (2013). *Mining Software Repositories to Acquire Software Risk Knowledge*. Paper read at the 14th International Conference on Information Reuse and Integration (IRI), San Francisco, California, USA.

Chau, T., Maurer, F., & Melnik, G. (2003). Knowledge sharing: Agile methods vs. tayloristic methods. *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Linz, Austria.

Charette, R.N. (2005). Why software fails. *IEEE Spectrum*, 42-49.

Chen, S. (2005). Task partitioning in new product development teams: A knowledge and learning perspective. *J. Eng. Technol. Manage*. 22, 291–314.

Chen, J., Tong, L., & Ngai, E.W.T. (2007). Inter-organizational knowledge management in complex products and systems: Challenges and an exploratory framework. *Journal of Technology Management in China*, 2(2), 134-144.

Cho, J.Y. & Lee, E.-H. (2014). Reducing confusion about grounded theory and qualitative content analysis: similarities and differences. *The Qualitative Report*, 19(32), 1-20.

Chouseinoglou, O. & Bilgen, S. (2012). A model for assessing organizational learning in software development organizations, in M. Winckler, P. Forbrig, and R. Bernhaupt (Eds.): *HCSE 2012, LNCS 7623*, pp. 251–258, 2012.

Chouseinoglou, O., İren, D., Karagöz, N.A. & Bilgen, S. (2013). AiOLoS: A model for assessing organizational learning in software development organizations. *Information and Software Technology*, 55(11), 1904–1924.
Chow, K.O., & Wong, T.L. (2011). A multiplicity approach to organization of knowledge and development of web software. *Proceedings of the Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing*, Sydney, Australia.

Chua, W.F.C. (1986*).* Radical developments in accounting thought. *The Accounting Review*, 61(4), 601-632.

Clerc, V., Lago, P., & van Vliet, H. (2011). Architectural knowledge management practices in agile global software development. *Paper read at the Sixth International Conference on Global Software Engineering Workshop*, Helsinki, Finland.

Coakes, E. (2006) Storing and sharing knowledge: Supporting the management of knowledge made explicit in transnational organisations. *The Learning Organization*, 13(6), 579-593.

Cobb, C.G. (2011). *Making sense of agile project management: balancing control and agility*. Hoboken, NJ.: John Wiley and Sons Inc.

Cohen, M.D. (1991). Individual learning and organizational routine: emerging connections. *Organisation Science*, 2(1), 135-139.
Collier, B., Demarco, T., & Fearey, P. (1996). A defined process for project post-mortem review. *IEEE Software*, 13, 65-73.

Collis, J., & Hussey, R. (2003). *Business research: A practical guide for undergraduate and postgraduate students* (2nd ed.). London: Palgrave MacMillan.

Corley, K.G., & Gioia, D.A. (2011). Building theory about theory building: what constitutes a theoretical contribution. *Academy of Management Review*, 36(1), 12–32.

Cresswell, J.W. (2009) *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (3rd Ed.), Los Angeles: SAGE.

Dalkir, K. (2011). *Knowledge management in theory and practice (2nd Ed.*). Cambridge: The MIT Press.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3), 319-339.

de Souza, E.F., Falbo, R d-A., & Vijaykumar, N.L. (2014). Knowledge management initiatives in software testing: A mapping study. *Information and Software Technology* xxx (2014) xxx–xxx.

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.

Deetz, S. (1996). Describing differences in approaches to organization science: rethinking Burell and Morgan and their legacy. Organization Science, 7 (2), 191-207.

Desouza, K.C., Awazu, Y., & Wan, Y. (2006). Factors governing the consumption of explicit knowledge. *Journal of the American Society for Information Science and Technology*, 57(1), 36–43.

Desouza, K.C., Dingsøyr., T., & Awazu, Y. (2005). Experiences with conducting project Postmortems: Reports versus stories. *Software Process Improvement and Practice*, 10, 203–215.

Dey, P.K., Kinch, J., & Ogunlana, S.O. (2007). Managing risk in software development projects: a case study. *Industrial Management & Data Systems*, 107(2), 284 – 303.

DiBella, A.J (1995). Developing learning organisations: a matter of perspective. *ACAD MANAGE PROC*, 1, 287-290.

Dingsoyr, T. (2000). An evaluation of research on experience factory. Proc. Workshop Learning Software Organizations (PROFES 2000). Retrieved from http://www.idi.ntnu.no/~dingsoyr/paper/lso2000.html

Dingsoyr, T. (2000). An evaluation of research on experience factory. Proc. Workshop Learning Software Organizations (PROFES 2000), 55-66. Retrieved from http://www.idi.ntnu.no/~dingsoyr/paper/lso2000.html

Dingsøyr, T. (2005). Postmortem reviews: purpose and approaches in software engineering. *Information and Software Technology*, 47, 293–303.

Dingsøyr, T., & Šmite, D. (2014). Managing knowledge in global software development projects". *IT Pro*, 22-29.

Dingsøyr, T., Moe, N.B., & Nytrø, Ø. (2001). Augmenting experience reports with lightweight Postmortem Reviews. *Proceedings of Third International Conference on Product Focused Software Process Improvement*, Kaiserslautern, Germany.

Dorairaj, S. Noble, J. & Malik, P. (2012) Technical Report 12-08: Knowledge Management in Distributed Agile Software Development. Retrieved for http://ecs.victoria.ac.nz/foswiki/pub/Main/TechnicalReportSeries/ECSTR12-08.pdf.

Earl, M. (2001). Knowledge management strategies: toward a taxonomy. *Journal of Management Information Systems*, 18(1), 215-233.

Easterby-Smith, M. & Lyles, M.A. (2011). In praise of organizational forgetting. *Journal of Management Inquiry*, 20(3), 311-316.

Ebad, S.A. (2016). Influencing factors for IT software project failures in developing countries — A critical literature survey. *Journal of Software*, 11(11), 1145-1153.

Egbokhare, F.A. (2014). Causes of software/information technology project failures in Nigerian software development organisations. *African Journal of Computing and ICT*, 7(2), 107-110.

El Emam, K., & Koru, A.G. (2008). A replicated survey of IT software project failures. *IEEE Software*, 84-90.

Elfving, S., & Funk, P. (2006). Enabling knowledge transfer in product development and production – methods and techniques from artificial intelligence. *Paper presented at the 1st Nordic Conference on Product Lifecycle Management, Goteborg, Sweden*.

Elo, S. & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of Advanced Nursing*, 62(1), 107–115.

Evans, M. & Ali, N. (2013). Bridging knowledge management lifecycle theory and practice", *Paper read at the International Conference on Intellectual Capital*, Tel-Aviv, Israel.

Ewusi-mensah, K. & Przasnyski, Z. H. (1991). On information systems project abandonment - An exploratory study of organizational practices. *MIS Quarterly*, 15(1), 67-86.

Ewusi-Mensah, K. (2003). Software development failures: anatomy of abandoned projects. Cambridge: MIT Press.

Ezeh, P.A.A. (2013). Factors influencing knowledge sharing in software development: a case study at Volvo Cars IT Torslanda (Unpublished Bachelor of Science Thesis) Department of Computer Science and Engineering Göteborg, Sweden.

Facin, A.L., Laurindo, F.J.B., & Spinola, M.M (2014). Knowledge management to deal with risk in the process of software development: A case study. *Proceedings of PICMET: Infrastructure and Service Integration*, Kanazawa, Japan.

Farenhorst, K., Izaks, R., Lago, P., & van Vliet, H. (2008). A just-in-time architectural knowledge sharing portal. *Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture*, Vancouver, Canada.

Farsi, J.Y. & Toghraee, M.T. (2014). Identification the main challenges of small and medium sized enterprises in exploiting of innovative opportunities (Case study: Iran SMEs). *Journal of Global Entrepreneurship and Research*, 4(4), 1-15.

Fassinger, R. and Marrow, S.L. (2013). Toward best practices in quantitative, qualitative, and mixed method research: a social justice perspective. *Journal for Social Action in Counselling and Psychology*, 5(2), 69-83.

FDI Intelligence (2012). Software & IT services: sector fact sheet. Retrieved from wesgro.co.za/publications/publication/software-and-it-services

Feher, P. & Gabor, A. (2006). The role of knowledge management supporters in software development companies, *Software Process Improvement and Practice*, 11(3), 251-260.

Feldmann, R.L., & Althoff, K-D. (2001) On the status of learning software organisations in the year 2001, in: Proceedings of the Learning Software Organizations Workshop, Springer Verlag, Kaiserslautern, Germany, 2001, pp. 2–6.

Ferguson, S. (2004). The knowledge management myth: will the real knowledge managers please step forward? Paper presented at ALIA 2004. Retrieved from http://conferences.alia.org.au/alia2004/pdfs/ferguson.s.paper.pdf.

Ferreira, C. & Cohen, J. (2008). Agile systems development and stakeholder satisfaction: A South African empirical study, in Charmain Cilliers, Lynette Barnard and Reinhardt Botha (Eds.) *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, Wilderness, South Africa.

Fiol, C.M., & Lyles, M.A. (1985). Organisational learning. *Academy of Management Review,* 10(4), 803-814.

Fontaine, M., & Lesser, E. (2002). Challenges in managing organizational knowledge. *IBM Institute for Knowledge Based Organizations.* Retrieved from http://retawprojects.com/uploads/challenges_in_managing_organizational_knowledge.pdf

Gallagher, K.P. *et al.* (2011). A typology of requisite skills for information technology professionals. *Proceedings of the 44th Hawaii International Conference on System Sciences* – Hawaii, USA.

Garcia, J., Amescua, A., Sanchez, M-I. & Bermon, L. (2011). Design guidelines for software processes knowledge repository development. *Information and Software Technology*, 53(8), 834-850.

Geethalakshmi, S.N. & A. Shanmugam, A. (2008). Success and failure of software development: practitioners' perspective. *Proceedings of the International Multi-Conference of Engineers and Computer Scientists,* Vol I, Hong Kong, China.

Gendreau, O. & Robillard, P.N. (2013). Knowledge acquisition activity in software development. In A. Rocha et al. (Eds.), *Advances in Information Systems and Technologies,* 206, 1–10, Springer-Verlag, Berlin.

Gladstone, B. (2000). *From know-how to knowledge: the essential guide to understanding and implementing knowledge management.* London: Industrial Society.

Glass, R.L. (2002). Project retrospectives, and why they never happen. *IEEE Software*, 110-112.

Gorman, G.E. (2004). The uniqueness of knowledge management – or the emperor's new clothes? Retrieved from http://www.emeraldinsight.com/librarians/info/viewpoints/unique.htm?view=print&PHPSESSID=pu41r40dt43fi1lvdtrre29ui0.

Gottschalk, P. (2007). *Knowledge management systems: value shop creation*. Hersey: IDEA group publishing.

Gourlay, S. (2006). Conceptualizing knowledge creation: A critique of Nonaka's theory. *Journal of Management Studies*, 43:7, 1415-1436.
Grant, K. (2015). Knowledge management: an enduring but confusing fashion, in Ken Grant, K. and John Dumay (eds), *Leading issues in knowledge management: for researchers, teachers, and students,* Vol 2, Reading: Academic Conferences, and Publishing International Limited.

Green, D. & David, J.L. (1984). A research design for generalising from multiple case studies. *Evaluation and Program Planning*, 7, 73-85.

Greene, J.C., Caracelli, V.J. & Graham, W.F. (1989). Toward a conceptual framework for mixed-method evaluation designs. *Educational Evaluation and Policy Analysis*, 11(3), 255-274.

Gruner, S., & van Zyl, J. (2011). Software testing in small IT companies: a (not only) South African problem. *South African Computer Journal*, 47, 7-32.

Gupta, B., Iyer, L.S., & Aronson, J.E. (2000). Knowledge management: practices and challenges. *Industrial Management & Data Systems*, 100(1), 17-21.

Habra, N. *et al*. (2008). Initiating software process improvement in very small enterprises Experience with a light assessment tool. *Information and Software Technology*, 50, 763–771.
Hackbarth, G. (1998). The impact of organizational memory on IT systems. AMCIS 1998 Proceedings. Paper 197. Retrieved from http://aisel.aisnet.org/amcis1998/197.

Hammond, M. & Wellington, J. (2013). *Research methods: key concepts*. London: Routledge Taylor and Francis.

Handzic, M. (2001**).** Knowledge management: a research framework, in D. Remenyi (ed.), *Second European Conference on Knowledge Management* (pp. 219-229), Bled, Slovenia.

Hansen, M.T., Nohria, N., & Tierney, T.J. (1999). What's your strategy for managing knowledge? Retrieved from https://hbr.org/1999/03/whats-your-strategy-for-managing-knowledge/ar/1
Harraf, A., Soltwisch, B.W. & Talbot, K. (2016). Antecedents of organizational complacency: identifying and preventing complacency in the work environment, *Managing Global Transitions,* 14(4): 385–40.

Hastie, S. & Wojewoda, S. (2015). Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch. Retrieved from http://www.infoq.com/articles/standish-chaos-2015

Heavin, C. & Adams, F. (2012). Characterizing the knowledge approach of a firm: An investigation of knowledge activities in five software SMEs", *The Electronic Journal of Knowledge Management*, 10(1), 48-63.

Heeks, R (2008). Success and failure rates of eGovernment in developing/transitional countries: overview. Retrieved from http://www.egov4dev.org/success/sfrates.shtml accessed 17 sept 2017.

Heeks, R. (2002). Information systems and developing countries: failure, success, and local improvisations. *The Information Society*, 18, 101-112.

Heisig, P. (2009). Harmonisation of knowledge management – comparing 160 KM frameworks around the globe. *Journal of Knowledge Management*, 13(4), 4 – 31.

Hey J. (2004). The data, information, knowledge, wisdom chain. Retrieved from http://scholar.google.co.za/scholar?q=The+Data,+Information,+Knowledge,+Wisdom+Chain+%2Bhey+2004&hl=en&um=1&ie=UTF-8&oi=scholart

Herrmann, v-N. (2011). Barriers for an efficient management of knowledge. Retrieved from http://www.community-of-knowledge.de/beitrag/barriers-for-an-efficient-management-of-knowledge/

Highsmith, J. & Cockburn, A. (2001). Agile software development: The business of innovation. *Software Management*, 120-122.

Hirschheim, R., & Klein, H.K. (1989). Four paradigms of information systems development. *Communication of the ACM*, 32(10), 1-15.

Hodge, G. (2000). *Systems of knowledge organization for digital libraries: beyond traditional authority files*. Washington: The Digital Library Federation Council on Library and Information Resources.

Holsapple, C.W., & Joshi, K.D. (2002). Knowledge management: a threefold framework. *The Information Society*, 18, 47-64.

Holsapple, C.W., & Singh, M. (2001). Knowledge chain model: Activities for competitiveness. *Expert Systems with Applications*, 20(1):77–98.
Holsapple, C.W., Jones, K., & Leonard, L.N.K. (2015). Knowledge acquisition and its impact on competitiveness. *Knowledge and Process Management*, 22(3), 157–166.

Holz, H., and Maurer F. (2002). Knowledge management support for distributed agile software processes. *Lecture Notes in Computer Science*, 2640:60-80.

Hoppe, A., Seising, R.., Nurnberger A., & Wenzel, C. (2011). Wisdom - the blurry top of human cognition in the DIKW-model? *EUSFLAT-LFA*, 584-591.

Huber, G.P. (1991). Organizational learning: the contributing processes and the literatures. *Organisation Science*, 2(1), 88-115.

Hughes, D.L., Rana, N.P., & Simintiras, A.C. (2017). The changing landscape of IS project failure: an examination of the key factors. *Journal of Enterprise Information Management*, 30(1), 142-165.

Humphrey, W. (1992) *Introduction to software process improvement* (Unpublished Technical Report CMU/SEI-92 TR007) Pittsburgh Software Engineering Institute, USA.

Huntley, S.L. (2003). Organizational learning in open-source software projects: An analysis of debugging data. *IEEE Transactions on Engineering Management*,50(4), 485-493.

Hussey, J., & Hussey, R. (1997). *Business research*: *A practical guide for undergraduate and postgraduate students*. London: McMillian Publishers.

Independent Communications Association of South Africa (2016). Report on the state of the ICT sector in South Africa. Retrieved from https://www.ellipsis.co.za/wp-content/uploads/2017/05/ICASA-Report-on-State-of-SA-ICT-Sector-2017.pdf

ITWeb Surveys: software development survey (2011); retrieved from http://www.itweb.co.za/index.php?option=com_content&view=article&id=47732&Itemid=2488

ITWeb Surveys: software development survey (2012); retrieved from http://www.itweb.co.za/index.php?option=com_content&view=article&id=60527&Itemid=2826

ITWeb Surveys: Software Development Survey (2013); retrieved from http://www.itweb.co.za/index.php?option=com_bfsurvey_pro&view=stats&catid=838&Itemid=2879.

Jansma, P.A. & Means, E.R. (2012). Developing the NASA systems engineering body of knowledge. *Aerospace Conference*, *Big Sky*, 1–17.

Jelínek, B.P. (2010). Knowledge management in software development (Unpublished Masters thesis), Masaryk University, Brno, Czech Republic.

Jiang, M., Jong, C., Poppell, P., Budhathoky, K., & Hull, R. (2009). Establishing business oriented system infrastructure development life cycle for enterprise systems. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.516.2335&rep=rep1&type=pdf

Johannesburg Centre for Software Engineering Source Code Report (2012). Retrieved from http://www.jcse.org.za.

Johnson, R.B. & Onwuegbuzie, A.J. (2004). Mixed methods research: a research paradigm whose time has come. *Educational Researcher*, 33(7), 14-26.

Jones, C. (1995). Patterns of large software systems: failure and success. *Computer*, 28(3), 86-87.

Jørgensen. M. & Moløkken-Østvold, K. (2006). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, 48, 297–301.

Jørgensen. M. (2014). Failure factors of small software projects at a global outsourcing marketplace. *The Journal of Systems and Software*, 92 (2014) 157–169.

Kalkan, V.D. (2008). An overall view of knowledge management challenges for global business. *Business Process Management Journal*, 14(3), 390-400.

Kamunya, S., & Waweru, M. (2013). Utilization of knowledge management tools in software development. *Paper read at the IST-Africa Conference and Exhibition*, Nairobi, Kenya.

Kappelman, L.A., McKeeman, R., and Zhang, L. (2006) Early warning signs of it project failure: The dominant dozen. *Information Systems Management*, 23(4), 31-36.

Karhu, K., Taipale, O., & Smolander, K. (2009). Investigating the relationship between schedules and knowledge transfer in software testing. *Information and Software Technology*, 51, 663–677.

Kasvi, J.J.J., Vartiainen, M. & Hailikari, M. (2003). Managing knowledge and knowledge competences in projects and project organisations. *International Journal of Project Management*, 21, 571–582.

Kaur, R. & Sengupta, J. (2011). Software process models and analysis of failure of software development projects. *International Journal of Scientific and Engineering Research*, 2(2), 1-4.

Keegan & Turner (2001). Quantity versus quality in project-based learning practices. *Management Learning*, 32(1), 77-98.

Keil, M. (1995). Pulling the plug: Software project management and the problem of project escalation. *MIS Quarterly*, 19(4), 421-447.

Kess, P., & Haapasalo, H. (2002). Knowledge creation through a project review process in software production. *International Journal of Production Economics*, 80(1), 49-55.

Khoza, L.T. & Pretorius, A.B. (2016). Knowledge sharing in software development projects. *Proceedings of the 13th The International Conference on Intellectual Capital, Knowledge Management and Organisational Learning*, Ithaca, NY, USA.

Kim, G., Lee, M., Lee, J., & Lee, K. (2005). Design of SPICE experience factory model for accumulation and utilization of process assessment experience. *Proceedings of the 3rd ACIS Int'l Conference on Software Engineering Research, Management and Applications*, Michigan, USA.

King, R.W. (2009). Knowledge management and organisational learning. *Annuals of Information Systems*, 4, 3-13.

Kiniti, S., & Standing, C. (2013). Wikis as knowledge management systems: issues and Challenges. *Journal of Systems and Information Technology*, 15(2), 189-201.

Klein, H.K., & Myers, M.D. (1999). A set of principles for conducting and evaluating interpretive field studies in Information systems. *MIS Quarterly*, 23(1), 67-94.
Klint, P. & Verhoef, C. (2002). Enabling the creation of knowledge about software assets. *Data & Knowledge Engineering*, 41(2-3), 141-158.

Kneale, L. (2017). *The information technology industry*. Johannesburg: Copyright Who Owns Whom (Pty) Ltd.

Krippendorff, K. (2004). *Content analysis: an introduction to its methodology*. London: SAGE Publications.
Kristjansson, B., Helms, R. & Brinkkemper, S. (2014). Integration by communication: knowledge exchange in global outsourcing of product software development. *Expert Systems,* 31(3), 267-280.

Kukko, M. (2013). Knowledge sharing barriers in organic growth: A case study from a software company. *Journal of High Technology Management Research*, 24, 18–29.

Kukko, M., Helanda, N., & Virtanen, P. (2008). Knowledge management in renewing software development processes. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, (pp332-342), Washington DC, IEEE Computer Society.

Kumarawadu, P. (2008). Achieving competitive advantage through knowledge management initiatives in small and medium software industry. *Journal of Information & Knowledge Management*, 7(4), 255–265.

Kuusinen K., Gregory P., Sharp H., Barroca L., Taylor K., & Wood L. (2017) Knowledge sharing in a large agile organisation: A survey study, in Baumeister H., Lichter H., Riebisch M. (eds) *Agile Processes in Software Engineering and Extreme Programming. XP 2017*, Lecture Notes in Business Information Processing, vol 283. Springer, Cham

Labuschagne, L., Marnewick, C., & Jakovljevic, M. (2008). IT project management maturity: A South African perspective. *Proceedings of the PMSA conference: from strategy to reality*, Midrand, South Africa.

Lakalu, M. (2010). A framework of collaborative knowledge management system in open source software development environment. *Computer and Information Science*, *3(*1), 81-90.
Lamoreux, M. (2005). Improving agile team learning by improving team reflections. *Proceedings of the Agile Development Conference (ADC'05),* IEEE Computer Society Washington, DC, USA.

Le Compte, M., & Goetz, J.P. (1982). Problems of reliability and validity in ethnographic research. *Review of Educational Research*, 52(1), 31-60.

Lee, A.S. (1989). A scientific methodology for MIS case studies. *MIS Quarterly*, 13(1), 33-50.

Lee, A.S. (1991). Integrating positivist and interpretive approaches to organisational research. *Organisational Science,* 2(4), 242-365.

Lee, C.K. & Han, H-J. (2008). Analysis of skills requirement for entry-Level programmer/analysts in fortune 500 corporations. *Journal of Information Systems Education,* 19(1), 17-27.

Lee, G.K. & Cole, R.E. (2003). From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. *Organisation Science*, 14(6), 633–694.

Lehtinen, T.O.A., Mäntylä, M.V., Vanhanen, J., Itkonen, J. & Lassenius, K. (2014). Perceived causes of software project failures – An analysis of their relationships. *Information and Software Technology,* 56, 623–643.

Leonard, D., & Sensiper, S. (1998). The role of tacit knowledge in group innovation. *Californian Management Review,* 40 (3), 112 – 132.

Lethbridge, T.C. (2000). What knowledge is important to a software professional? *Computer,* 44-50.

Li, M & Gao, F. (2003). Why Nonaka highlights tacit knowledge: a critical review. *Journal of Knowledge Management,* 7(4), 6-14.

Liao, Y. & Marsillac, E. (2015). External knowledge acquisition and innovation: the role of supply chain network-oriented flexibility and organisational awareness. *International Journal of Production Research,* 53(18), 5437– 5455.

Liebenberg, J., Huisman, M., & Mentz, E. (2014). Knowledge and skills requirements for software developer students. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering,* 8(8), 2604-2609.

Linberg, K.R. (1999). Software developer perceptions about software project failure: a case study. *The Journal of Systems and Software*, 49, 177-192.

Linden, T. & Cybulski, J. (2009). Knowledge creation in an application domain: A hermeneutic study in ICKM 2009. *Proceedings of the 6th International Conference on Knowledge Management: Managing knowledge for global and collaborative innovations*, Hong Kong, China.

Lindvall, M., Rus, I., & Sinha, S. (2002). Technology support for knowledge management. Fraunhofer Center for Experimental Software Engineering, College Park, and the Department of Computer Science, University of Maryland at College Park.

Liu, Y., Wu, J., Liu, X., & Gu, G. (2009). Investigation of knowledge management methods in software testing process. In *2009 International Conference on Information Technology and Computer Science*, Kiev, Ukraine, IEEE Computer Society.

Lyytinen, K. & Hirschheim, R. (1987) Information systems failures: A survey and classification of the empirical literature. *Oxford Surveys in Information Technology*, 4, 257-309.

Lyytinen, K. (1988). Expectation failure concept and systems analysts' view of information systems failures: results of an exploratory study. *Information and Management*, 14, 25-56.

Lyytinen, K., & Robey, D. (1999). Learning failure in information systems development. *Information Systems Journal*, 9, 85-101.

Ma, R. & Huang, Y-C. (2016). Opportunity-based strategic orientation, knowledge acquisition, and entrepreneurial alertness: The perspective of the global sourcing suppliers in China. *Journal of Small Business Management*, 54(3), 953–972.

Maggs, J. (2010). SMMEs' centre stage in the SA economy. *Professional Accountant*, 1, p.5.

Maglyas, A. (2010). Evaluating the success of software development projects in Russia, Ukraine and Belarus (Unpublished Masters Dissertation), Lappeenranta University of Technology, Finland.

Maher, P.E., Kourik, JL. & Chookittikul, W. (2010). Exploratory study of agile methods in the Vietnamese software industry. *Proceedings of the Fifth International Multi-conference on Computing in the Global Information Technology*, Valencia, Spain.

Major, E., & Cordey-Hayes, M. (2000). Knowledge translation: a new perspective on knowledge transfer and foresight. *The Journal of Future Studies, Strategic Thinking and Policy*, 2(4), 413-423.

Maki, E., Jarvenpaa, E., & Hamalainen, L. (2001). Managing knowledge processes in knowledge intensive work, in D. Remenyi (ed.), *Second European Conference on Knowledge Management* (pp. 313-319), Bled, Slovenia.

Manifesto for Agile Software Development (2001). Retrieved from http://agilemanifesto.org/

Marsh, D. & Furlong, P. (2002) A skin not a sweater: ontology and epistemology in Political Science, in D. Marsh and G. Stoker (Eds) *Theory and Methods in Political Science* (2Ed) Chapter 1, pp. 17-11.

Marwick, A.D. (2001**).** Knowledge management technology**.** *IBM Systems Journal*, 40(4), 814-830.

Masinde, D. (2016). Why M-Pesa is a success story in Kenya. Retrieved from https://www.news24.com/Africa/News/why-m-pesa-is-a-success-story-in-kenya-20160517

McLeod, L., & McDonell, S.G. (2011) Factors that affect software systems development project: a survey of research, *ACM Computing Survey*, 43(4), 24-56.

Mat, A.R. (2011). Organizing knowledge to support requirements analysis. *Proceedings of the 3rd International Conference on Computer Research and Development*. Shanghai, China.

Mat, A.R., & Liu, S. (2011). Organizing knowledge to support requirements analysis. *Proceedings of the 3rd International Conference on Computer Research and Development (ICCRD)*, Shanghai, China.
Mathiassen, L., & Pourkomeylian, P. (2003). Managing knowledge in a software organization. *Journal of Knowledge Management, 7*(2), 63-80.
Mathrani, A., Parsons, D., & Mathrani, S. (2012). Knowledge management initiatives in off-shore software development: vendors' perspectives. *Journal of Universal Computer Science*, 18(19), 2706-2730.

May, L.J. (1998) Major causes of software project failures. *CROSSTALK The Journal of Defense Software Engineering*, 9-12.

McElroy, M. (2003). *The new knowledge management: complexity, learning and sustainable innovation*, Boston: Butterworth-Heinemann.

McGiven, Y. (2006). *The practice of Marketing and Social Research: An Introduction* (2nd ed.). England: Prentice Hall.

McGrath, K. (2005). Doing critical research in information systems: A case of theory and practice not informing each other. *Information Systems Journal*, 15, 85-101.

McManus, J. & Wood-Harper, T. (2007). Understanding the sources of information systems project failure: A study in IS project failure, *Management Services*, 38-43.

Mehta, N. (2008). Successful knowledge management implementation in global software companies. *Journal of Knowledge Management*, 12(2), 42 – 56.
Mendez, M.M., Gomes, J.F.S., & Batiz-Lazo B. (2004). Management of knowledge in new product development in Portuguese higher education, in J.N.D. Gupta & S.K. Sharma, *creating knowledge based organisations* (pp. 149-168), London: Idea Group Inc.

Menolli, A. L., Malucelli, A. & Reinehr, S. (2011). Towards a semantic social collaborative environment for organizational learning. *Proceedings of the 7th International Conference on Information Technology and Applications (ICITA 2011)*, Sydney, Australia

Mestad, A., Myrdal, R., Dingsøyr, T., & Dybå, T. (2007). Building a learning organization: Three phases of communities of practice in a software consulting company. *Proceedings of the 40th Hawaii International Conference on System Sciences*, Hawaii, USA.

Meyer, M. & Zack, M. (1996). The design and implementation of information products. *Sloan Management Review*, 37(3), 43-59.

Mishra, D., & Mahanty, B. (2015). Business knowledge requirements and onsite offshore work division in Indian software outsourcing projects. *Strategic Outsourcing: An International Journal*, 8(1), 76-101.
Mitchell, R., & Boyle, B. (2010). Knowledge creation measurement methods. *Journal of Knowledge Management*, 14(1), 67-82.

Moniruzzaman, A.B.M. & Hossain, S.A. (2013). Comparative study on agile software development methodologies. *Global Journal of Computer Science and Technology Software & Data Engineering*, 13(7).

Morner, M. & Von Krogh, G. (2009). A note on knowledge creation in open-source software projects: What can we learn from Luhmann's theory of social systems. *Systems Practice and Action Research*, 22(6), 431–443.

Morse, J. M., Barrett, M., Mayan, M., Olson, K., & Spiers, J. (2002). Verification strategies for establishing reliability and validity in qualitative research. *International Journal of Qualitative Methods 1* (2), Article 2. Retrieved from http://www.ualberta.ca/~ijqm/

Mukherji, S. (2004). Knowledge management strategy in software services organisations: straddling codification and personalisation. *IIMB Management Review*, 33-39.
Mutula, S.M. & van Brakel, P. (2006). E-readiness of SMEs in the ICT sector in Botswana with respect to information access. *The Electronic Library*, 24(3), 402-417.

Myers, M.D. (2013). *Qualitative research in business and management* (2nd Ed.). Los Angeles: SAGE Publications Inc.

Nahapiet, J. & Ghoshal, S. (1998). Social capital, intellectual capital, and organisational advantage. *Academy of Management Review*, 23(2), 242-266.

Nauman, A.B., Aziz, R., & Ishaq, A.F.M. (2005). Information systems development failure: A case study to highlight the IS development complexities in simple, low risk projects in developing countries. *Proceeding of the Second International Conference on Innovations in Information Technology*, Dubai: UAE.

Nawina, D.P. (2011). A model of knowledge management: delivering competitive advantage to small & medium scale software industry in Sri Lanka. *Proceedings of the 6th International Conference on Industrial and Information Systems*, Sri Lanka.

Ndiaye, O.K. (n.d.). Is the success of M-Pesa empowering Kenyan women? Retrieved from http://agi.ac.za/sites/agi.ac.za/files/standpoints_is_the_success_of_m-pesa_empowering_kenyan_rural_women_.pdf

Nesheim, T., Olsen, K.M., & Tobiassen, A.E. (2011). Knowledge communities in matrix-like organizations: managing knowledge towards application. *Journal of Knowledge Management*, 15(5), 836-850.

Neto, F.S. (2016). Impact of agile practices on organization learning: a model for knowledge creating and sharing in agile teams (Unpublished Masters Dissertation), Sistemas de Informação e Gestão do Conhecimento of FUMEC University.

Neves, F.T., Correia, A.M.R., Rosa, V.N. & Neto, M. (2011). Knowledge creation and sharing in software development teams using agile methodologies: Key insights affecting

their adoption. *Proceedings of the 6th Information Systems and Technologies Iberian Conference (CISTI)* (pp. 1-6), Chaves, Portugal.

Neves, S.M. *et al*. (2014). Risk management in software projects through knowledge management techniques: Cases in Brazilian incubated technology-based firms. *International Journal of Project Management*, 32, 125–138.

Ng, J.C.Y. (2012). Organizational learning has occurred! Or is it? *Review of Business Research*, 12(1), 121-131.

Niazi, M. (2009). Software process improvement implementation: Avoiding critical barriers. *CROSSTALK The Journal of Defence Software Engineering, 24-27.*

Nickols, F. W. (2000) The knowledge in knowledge management, in J.W. Cortada & J.A. Woods (eds.). *The knowledge management yearbook 2000-2001* (pp. 12-21). Boston, MA: Butterworth-Heinemann.

Nidhra, S., Yanamadala, M., Afzal, W., & Torkar, R. (2013). Knowledge transfer challenges and mitigation strategies in global software development—A systematic literature review and industrial validation. *International Journal of Information Management*, 33, 333– 355.

Nieuwenhuis, J. (2007). Qualitative research designs and data gathering techniques. In Kobus Maree (Ed.), *First steps in research* (revised edition), Pretoria: Van Schaik Publishers.

Noble, H., & Smith, J. (2015). Issues of validity and reliability in qualitative research. *Evid Based Nurs,* 18(2), 34-35.

Nogeste K., & Walker, D.H.T. (2006). Using knowledge management to revise software-testing processes. *Journal of Workplace Learning*, 18(1), 6 – 27.

Nonaka, I. (1994). A dynamic theory of organisational knowledge creation, *Organisation Science*, 5(1), 14-37.

Nonaka, I. & Takeuchi, H. (1995). *The knowledge-creating company*. New York: Oxford University Press.

Nonaka, I. (1997). Organisational knowledge creation. Retrieved from http://www.knowledge-nurture.com/downloads/NONAKA.pdf

Nonaka, I., & Konno, N. (1998). The concept of "Ba": building a foundation for knowledge creation. *California Management Review*, 40(3), 40-54.

Noruwana, N., & Tanner, M. (2012). Understanding the structured processes followed by organisations prior to engaging in agile processes: A South African Perspective. *South African Computer Journal*,48, 41-58.

Ocholla, D., & Shongwe, M. (2013). An analysis of the LIS job market in South Africa. *South African Journal of Libraries and Information Science*, 79(1). DOI: http://dx.doi.org/10.7553/79-1-113.

O'dell, C., Grayson, C. J., & Essaides, N. (2003). If only we knew what we know: The transfer of internal knowledge and best practice. New York: Free Press.

O'Neil, S. (2005). Managing tacit knowledge in a Hi-Tech learning organisation (Unpublished Masters Dissertation), Durban University of Technology, South Africa.

Olikowsky, W. J. & Baroudi, J.J. (1991). Studying information technology in organisations: Research approaches and assumptions. *Information Systems research,* 2(1), 1-28.

Olivera, F. (2000). Memory systems in organizations: an empirical investigation of mechanisms for knowledge collection, storage and access. *Journal of Management Studies*, 37(6), 811-832.

Organisation for Economic Cooperation and Development (OECD) (2017). Enhancing the contributions of SMEs in a global and digitalised economy. Retrieved from https://www.oecd.org/mcm/documents/C-MIN-2017-8-EN.pdf

Örtenblad, A. (2001). On differences between organizational learning and learning organization. *The Learning Organization,* 8(3), 125-133.

Paasivaara, M. & Lassenius, C. (2006). Could global software development benefit from agile Methods? *Paper presented at the IEEE International Conference on Global Software Engineering*, Florianopolis, Brazil.

Paasivaara, M. & Lassenius, C. (2014). Communities of practice in a large distributed agile software development organization – case Ericson. *Information and Software Technology*, 56 1556–1577.

Palvia, P., Mao, E.N., Salam, AF., & Soliman, K.S. (2003). Management information systems research: what's there in A methodology? *Communications of the Association for Information Systems,* 11, 289-309.

Parviainen, P., & Tihinen, M. (2014). Knowledge-related challenges and solutions in GSD. *Expert Systems*, 31(3), 253-266.

Pástor, R., Šipikal, M. & Rehák, S. (2013). Knowledge creation and knowledge acquisition in the software industry in Slovakia: The case study of Košice region. *Regional Science Policy & Practice* 5(4), 401–416.

Paulin, D., & Suneson, K. (2012). Knowledge transfer, knowledge sharing and knowledge barriers—Three blurry terms in KM. *Electronic Journal of Knowledge Management,* 10(1), 81-91.

Peler, M., Boydell, T., & Burgoyne, J. (1989). Towards the learning company. *Management learning,* 20(1), 1-8.

Peng, G., & Mu, J. (2013). Learning and open source software license choice. *Decision Sciences,* 44(4), 619-643.

Pentland, B.T. (1995). Information systems and organizational learning: the social epistemology of organizational knowledge systems. *Accounting, Management and Information Technologies,* 5(1), 1995, 1-21.

Peters, R. & Naicker, V. (2013). Small medium micro enterprise business goals and government support: A South African case study. *South African Journal of Business Management*,44(4), 13-24.
Pinjani, P., & Palvia, P. (2013). Trust and knowledge sharing in diverse global virtual teams. *Information and Management, 50*(4), 144-153.
Pinto, J.K. & Mantel, S.J. Jr. (1990). The causes of project failure. *IEEE Transactions on Engineering Management*, 37(4), 269-276.

Poell, R.F., & Van de Krogt, F.J. (2003). Learning strategies of workers in the knowledge-creating company. *Human Resource Development International*, 6(3), 387–403.
Polanyi, M. (1962). Tacit knowing: It's bearing on some problems of philosophy. *Reviews of Modern Physics*, 34(4), 601-616.
Ponelis, S.R. & Holmner, M.A. (2015). ICT in Africa: enabling a better life for all. *Information Technology for Development*, 21(1), 1-11.

Poon, P. & Wagner, C. (2001). Critical success factors revisited: success and failure cases of information systems for senior executives. *Decision Support Systems*, 30(4), 393–418.

Pretorius, S. Steyn, H. & Jordaan, J.C. (2012). Project management and project management success the engineering and construction industries in South Africa. *South African Journal of Industrial Engineering*, 23 (3), 1-12.

Proctor, T. (2005). *Essentials of Marketing Research* (4th ed.). England: Prentice Hall.

Rahman, R.A. (n.d.). Why projects fail. Retrieved from http://intosaiitaudit.org/muscat/Jordan-Why_IT_Projects_Fail.pdf 18 October, 2017.

Rajkumar, G. & Alagarsamy, K. (2013). The most common factors for the failure of software development project. *The International Journal of Computer Science & Applications*, 1(11),74-77.

Raju, B. (2012). Knowledge management and software development organizations, *Methods and Tools.* Retrieved from http://www.methodsandtools.com/archive/knowledgemanagement.php

Ranjbarfard, M., Aghdasi, M., López-Sáez, P., & López, J.E.N. (2014). The barriers of knowledge generation, storage, distribution and application that impede learning in gas and petroleum companies. *Journal of Knowledge Management*, 18(3), 494-522.

Ramnath, V. (2010). The level of adoption and effectiveness of software development methodologies in the software development industry in South Africa (unpublished Masters Dissertation), University of Pretoria, South Africa.
Ras, E. & Weber, S. (2009). Software organization platform: integrating organizational and individual learning. *Proceedings of Wikis4SE'*, Vancouver, Canada.

Rathi, D., Given, L.M., & Forcier, E. (2014). Interorganisational partnerships and knowledge sharing: the perspective of non-profit organisations (NPOs). *Journal of Knowledge Management*, 18(5), 867-885.

Raths, D. (2012). The ultimate KM challenge. *KMWorld* , 10-12.

Razzak and Ahmed (2014). Knowledge Sharing in Distributed Agile Projects: Techniques, Strategies and Challenges. *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems* (pp. 1431–1440).

Razak, N.A. *et al*. (2016). Theories of knowledge sharing behaviour in business strategy. *Procedia Economics and Finance*, 37, 545 – 553.

Retief, E. (2010). Business hurdles facing SMMEs. *Professional Accountant*, 1, 14 – 15

Rezende, M.S.C. & Alves, J. (n.d.). Evaluating knowledge management practices in software development organizations. Retrieved from http://www.convibra.com.br/dwp.asp?id=9228&ev=71.

Rice, J.L., & Rice, B.S. (2005). The applicability of the SECI model to multi-organisational endeavours: an integrated review. *International Journal of Organizational Behaviour*, 9(8), 671-682.

Richardson, I. & von Wangenheim, C.G. (2007). Why are small software organisations different? *IEEE Software*, 24(1), 18-22.

Robertson, P.L. (2003). The role of training and skilled labour in the success of SMEs in developing economies. *Education & Training*, 45(8/9), 461-473.

Robillard, P.N. (1999) The role of knowledge in software development. *Communications of the ACM*, 42(1), 87-92.

Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on agile and lean usage in Finnish software industry. *ESEM'12*, Lund, Sweden.

Rogers, E.M. (2003). *Diffusion of innovations* (5th ed.). New York: Free Press.

Rolfe, G. (2006). Validity, trustworthiness and rigour: quality and the idea of qualitative research. *Methodological Issues in Nursing Research*, 53(3), 304-310.

Rollett, H. (2003). Knowledge management processes and technologies. Kluwer: Norwell.

Rumizen, M.C. (2004). *The complete idiot's guide to knowledge management.* USA: CWL Publishing Enterprise.

Ryan, S., & O'Connor, R. V. (2013). Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology*, *55*(9), 1614-1624.

Sagsan, M. & Zorlu, K. (2010). An empirical test of the knowledge management life cycle model at a Turkish petroleum oil industry firm. *Proceedings the 7th International Conference on Intellectual Capital, Knowledge Management & Organisational Learning*, Hong Kong, China.

Sağsan, M. (2006). A new lifecycle model for processing knowledge management. *Paper presented at the 2nd International on Business, Management and Economics*, Izmir, Turkey, 2006.

Sağsan, M. (2009). Knowledge management discipline: test for an undergraduate program in Turkey. *Electronic Journal of Knowledge Management*, 7(5), 627–636.

Sharma, N., Singh, K. & Goyal, D.P. (2012) Adoption of knowledge management practices in software engineering organizations. In *2012 Second International Conference on Advanced Computing & Communication Technologies (*pp. 24–29*)*, Rohtak, India.

Salo O. & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. *IET Software*, 2(1), 58–64.

Sambamurthy, V., & Subramani, M. (2005). Special issues on information technologies and knowledge management. *MIS Quarterly,* 29 (1), 1-7.

Samoilenko, N. & Nahar, N. (2013). Knowledge sharing and application in complex software and systems development in globally distributed high-tech organisations using suitable IT tools. *Paper read at the Portland International Center for Management of Engineering and Technology (PICMET '13),* San Jose, USA.

Saranya, A., & Kannan, S. (2013). SPI challenges of small and medium sized software companies – problems and prospects. *International Journal of Engineering Research & Technology*, 2(1), 1-5.

Satzinger, J.W., Jackson, R.B., & Burd, S.D. (2014). *Introduction to systems analysis and design: an agile iterative approach.* Andover: Cangage Learning EMEA.

Saunders, M., Lewis, P., & Thornhill, A. (2009). *Research methods for business students* (5th Ed.). Harlow: Prentice Hall.

Savolainen, P., & Ahonen, J.J. (2015). Knowledge lost: Challenges in changing project manager between sales and implementation in software projects. *International Journal of Project Management*, 33(1), 92-102.

Scherp, A., Schwagereit, F., & Ireson, N. (n.d.). Web 2.0 and traditional knowledge management processes. Retrieved from http://ansgarscherp.net/publications/pdf/W13-ScherpSchwagereitIreson-Web2.0AndTraditionalKMProcesses.pdf

Schneider, K. (2009). *Experience and knowledge management in software engineering*. Hannover: Springer.

Schuh, K., & Barab, S.A. (2008). Philosophical perspectives. *In Handbook of Research on Educational Communications and Technology*, (3rd Ed.). J. Michael Spector, M. David Merrill, Jeroen van Merrienboer, and Marcy P. Driscoll. New York, Routledge Taylor & Francis.

Schultzer, U., & Leidner, D.E. (2002). Studying knowledge management in information systems research: discourses and theoretical assumptions. *MIS Quarterly*, 26(3), 213-242.

Seaman, C. B., Mendonca, M., Basili, V., & Kim, Y-M. (1999). An experience management system for a software consulting organization, Software Engineering Workshop, NASA/Goddard Software Engineering Laboratory, Greenbelt, MD, December 1999.

Segal, J. (2004). *Professional end user developers and software development knowledge* (Unpublished paper), The Open University, Milton Keynes, United Kingdom.

Senge, P. (1990), *The Fifth Discipline: The Art and Practice of the Learning Organisation*, Doubleday, New York, NY.

Senge, P.M., *et al*. (1994). *The fifth Discipline Fieldwork: Strategies and tools for building a Learning organization*. New York: Doubleday Inc. Publishers.

Serena (2007). *An Introduction to Agile Software Development*. California: San Mateo.

Sertić, M.B., Barčić, A.P., & Klarić, K. (2018). Economic determinants and analysis of the European Union wood industry SMEs employment. *BioResources*, 13(1), 522-534.

Sholla, A., & Nazari, E. (2011). Knowledge management and factors that influence the success of codification strategies in medium-sized companies that develop software: the model, strategies and tools. *Journal of Information Technology and Economic Development*, 2(1), 54-63.

Shongwe, M.M. (2015). Knowledge creation in students' software development teams. *South African Journal of Information Management*, 17(1). http://dx.doi.org/10.4102/sajim.v17i1.613

Shongwe M. (2016a). An analysis of knowledge management lifecycle frameworks: Towards a unified framework. *The Electronic Journal of Knowledge Management*, 14(3), 140-153.

Shongwe, MM. (2016b). Knowledge management in software development: a literature review. *Proceedings of the 13th International Conference on Intellectual Capital, Knowledge Management & Organisational Learning* – ICICKM 2016, Ithaca, NY, USA.

Sidenvall, A. (2017). *Knowledge sharing in and between agile so□ware development teams using knowledge processes: an interpretive case study at a medium-sized medical IT company* (Unpublished Masters Dissertation), Department of Management and Engineering, Linköping University.

Silverman, S. (2006). *Improving systems development methods by incorporating the principles of knowledge management* (Unpublished Masters Dissertation), University of Witwatersrand, South Africa.

Sivanantham, V. (2012). Knowledge management in agile projects. *Cognizant.* Teaneck.

Skyrme, D. J. (1998). Knowledge management solutions - the IT contribution. *ACM SIGGROUP Bulletin*, 19(1), 34-39.

Smit, W. (2017). SMMEs contribute 36% to economy. Retrieved from https://www.iol.co.za/business-report/opinion/smmes-contribute-36-to-economy-8269623

Smite, D. (2006). Requirements management in distributed projects. *Proceedings of the Workshop on Learning Software Organizations and Requirements Engineering*, University of Hannover, Germany.

Smith, J. (2002). A South African perspective on information technology software project failure. *Paper presented at African Rhythm Project Management Conference*, Johannesburg, South Africa

Smith, J. G., & Lumba, P. M. (2008). Knowledge management practices and challenges in international networked NGOs: the case of one world international. *The Electronic Journal of Knowledge Management,* 6(2), 167 – 176.

Soakell-Ho, M. & Myers, M.D. (2011). Knowledge management challenges for nongovernment organizations: The health and disability sector in New Zealand", *VINE*, 41(2), 212-228.

Song, M., Van der Bij, H., & Weggeman, M. (2005). Determinants of the level of knowledge application: a knowledge-based and information-processing perspective. *Journal of Product Innovation Management*, 22, 430-44.

Soundararajan, S. & Arthur, J.D. (2011). A structured framework for assessing the "goodness" of agile methods. *Paper presented at the 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, Las Vegas, Nevada, USA.

Spender, J.-C. (1996). Organizational knowledge, learning and memory: three concepts in search of a theory. *Journal of Organizational Change Management,* 9(1), 63-78.

Spender, J.-C. (2003). Knowledge fields: some post 9/11 thoughts about the knowledge – based theory of the firm. In C.W. Holsapple (ed.), *Handbook on Knowledge Management Volume 1*(pp. 59-71). Heidelberg: Springer –Verlag.

Stair, R. & Reynolds, G. (2012). *Information systems* (10th ed.). Australia: Course Technology Cengage Learning.

Statistics South Africa (2017). Information and Communication Technology satellite account for South Africa, 2013 and 2014. Retrieved from http://www.statssa.gov.za/publications/Report-04-07-01/Report-04-07-012014.pdf

Stein, E. W., & Zwass, V. (1995). Actualizing organizational memory with information systems. *Information Systems Research*, 6(2), 85-117.

Stenmark, D. (2001). Leveraging tacit organizational knowledge. *Journal of Management Information Systems,* 17(3), 9 – 24.

Surakka, S. (2007). What subjects and skills are important for software developers? *Communications of the ACM*, 50(1), 73-78.

Swart, M. (2010). Small businesses are set to lead economic recovery. *Professional Accountant*, 1, 10 – 12.

Switzerland Global Enterprise (2014) Information & communication technology (ICT) industry – South Africa; retrieved October 25, 2016 from http://www.s-ge.com/sites/default/files/private_files/1411_sA_ICT.pdf.

Tarawneh, H. (2011). A suggested theoretical framework for software project success. *Journal of Software Engineering and Applications*, 4, 646-651.

Tarawneh, M.M.I., Al Tarawneh, H., & Elsheikh, A. (2008). Software development projects: An investigation into the factors that affect software project success/ failure in Jordanian Firms. *Proceedings of the First International Conference on the Applications of Digital Information and Web Technologies*, Ostrava, Czech Republic.

Taweel, A., Delaney, B., & Zhao, L. (2009). Knowledge management in distributed scientific software development. *Proceedings of the Fourth IEEE International Conference on Global Software Engineering*, Limerick, Ireland.

The Bureau of Economic Research (2016). The small, medium and micro enterprise sector of South Africa. Retrieved from http://www.seda.org.za/Publications/Publications/The%20Small,%20Medium%20and%20Micro%20Enterprise%20Sector%20of%20South%20Africa%20Commissioned%20by%20Seda.pdf

The Department of trade and industry, republic of South Africa: annual review of small businesses in South Africa 2005-2007 (n.d.). Retrieved from http://www.thedti.gov.za/sme_development/docs/smme_report.pdf

The Standish Group CHAOS Manifesto (2015). Think Big, act small. Retrieved from https://www.standishgroup.com/b

The Standish Group CHAOS Manifesto: think big, act small (2013). Retrieved from www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf

The Standish Group Report (1995). CHAOS. Retrieved from https://net.educause.edu/ir/library/pdf/NCP08083B.pdf.

Tilley, S. & Rosenblatt, H. (2017). *Systems analysis and design* (11th ed.). Boston, MA: Cengage Learning Publishers.

Tiwana, A. (2002). *The knowledge management toolkit.* New Jersey: Pearson Education Inc.

Tiwana, A. (2004a). An empirical study of the effect of knowledge integration on software development performance, *Information and Software Technology*, 46, 899–906.

Tiwana, A. (2004b). Beyond the black box: knowledge overlaps in software outsourcing. *IEEE Software*, 21(5), 51-58.

Trimble, J.A. (2000). Structuring knowledge acquisition in software development projects, *SACJ / SART*, 26, 172-180.
Tsai, C.H., Chang, C.L., & Chen, L. (2006). A case study of knowledge management implementation for information consulting company. *International Journal of the Computer, the Internet and Management*, 14(3), 60-78.

Tsirakidis, P., Kobler, F., & Krcmar, H. (2009). Identification of success and failure factors of two agile software development teams in an open source organization. *Paper presented at the Fourth IEEE International Conference on Global Software Engineering*, Limerick, Ireland.

Tsoukas, H. (1996). The firm as a distributed knowledge system: a constructionist approach. *Strategic Management Journal*, 17, 11-25.

Tustin, M., Ligthelm, J.K., Martins,M., & van Syk, G. (2006). *Marketing Research in Practice*. Pretoria: Unisa Press.

United States International Trade Commission (2010). Small and medium-sized enterprises: overview of participation in U.S. exports. Retrieved from https://www.usitc.gov/publications/332/pub4125.pdf

Vaezi, S.K. (2005). Application of knowledge management in organizational management. *Proceedings of the Fifth International Conference on Electronic Business* (pp. 533 - 537), Hong Kong, China.

Valentino, M. (2017). Multiple case study. *Nature of Science: Research Methodologies, 1-4.*

Vasanthapriyan, S., Tian, J., & Xiang, J. (2015). A Survey on knowledge management in software engineering. *Paper read at the 2015 IEEE International Conference on Software Quality, Reliability and Security – Companion*, Vancouver, BC, Canada.

Venkatesh, V., Brown, S.A., & Bala, H. (2013). Bridging the qualitative–quantitative divide: guidelines for conducting mixed methods research in information systems. *MIS Quarterly*, 37(1), 21-54.
Venkatesh, V., Morris, M.G., Davis, G.B, & Davis, F.D. (2003) User acceptance of information technology: toward a unified view. *MIS Quarterly*, 27(3), 425-478.

VersionOne (n.d.). What Is Agile Methodology? Retrieved from https://www.versionone.com/agile-101/agile-methodologies/
Vijayasarathy, L. R., & Turk, D. (2008). Agile software development: A survey of early adopters. *Journal of Information Technology Management*, 19(2), 1-8.

Vithana, V.N., Fernando, S. G. S., & Kapurubandara, M. (2015). Success factors for agile software development – a case study from Sri Lanka. *International Journal of Computer Applications*, 113(17), 10-18.

Von Krogh, G. (1998). Care in knowledge creation. *California Management Review*, l (40), 133-154.

Walz, D.B., Elam, J.J. & Curtis, B. (1993). Inside a software design team: knowledge acquisition, sharing and integration. *Communications of the ACM*, 36(10), 63-77.

Wan, J., Zhang, H., Wan, D., & Huang, D. (2010). Research on knowledge creation in software requirement development. *Journal of Computing& Applications*, l3, 487-494.

Wenger, E. (1998). *Communities of practice: learning, meaning, and identity*. Cambridge: Cambridge University Press.
Wenger, E. (2009). Communities of practice: a brief introduction. Retrieved from http://www.ewenger.com/theory/

Wiig, K. (1993). *Knowledge management foundations: thinking about thinking, how people and organisations create, represent, and use knowledge.* Arlington: Schema Press.

Wilson, T.D. (2002). The nonsense of knowledge management. *Information Research*, 8(1), 1-35.

Wilson, T.D. (2005). The nonsense of knowledge management revisited. In Maceviciute, E. & Wilson, T.D. (Eds). *Introducing information management: an information researcher reader(pp.155-164)*. London: Facet Publishing.

Witherspoon, L.C., Jason, B., Cam, C., & Dan, S.N., (2013). Antecedents of business knowledge sharing: a meta-analysis and critique. *Journal of Knowledge Management*, 17(2), 250-277.

Xu, P. & Yao, Y. (2013). Knowledge sharing in offshore software development: a vendor perspective. *Journal of Global Information Technology Management*, 16(1), 58-84.

Yanghua, J. (2008). Collective goal orientation and justice climate on team learning. *Proceedings of the International Conference on Computer Science and Software Engineering*, Wuhan, Hubei, China.
Yeo, K.T. (2002). Critical failure factors in information system projects. *International Journal of Project Management.* 20, 241–246.

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. (1999). Enhancing the quality of case studies in health services research. *Health Services Research*, 34(5), 1209–1224.
Yin, R.K. (2009). *Case study research: design and methods* (4th Ed.). Los Angeles: SAGE Publications.

Yin, R.K. (2014). Case study research: design and methods (5th Ed.). Los Angeles: SAGE Publications Inc.

Yoshino, N. & Taghizadeh-Hesary, F. (2016). Major challenges facing small and medium-sized enterprises in Asia and solutions for mitigating them. (Unpublished Report), Asian Bank Institute, Tokyo, Japan.

Zahedi, M., Shahin, M., & Babar, M.A. (2016). A systematic review of knowledge sharing challenges and practices in global software development, *International Journal of Information Management*, 36, 995–1019.

Zander, U., & Kogut, B. (1995). Knowledge and the speed of the transfer and imitation of organisational capabilities: an empirical test. *Organisation Science*, 6(1), 76-92.

Zanello, G. & Maassen, P. (2011) Strengthening citizen agency and accountability through ICT. *Public Management Review*, 13(3), 363-382.

Zheng, Z. (2012). The design of technical systems for software project team sharing knowledge. Paper presented at the 24th Chines Control and Decision Conference, Taiyuan, China.

Zins, C. (2007). Conceptual approaches for defining data, information and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4):479–493.

Appendix A: Interview schedule for IT/project managers

Section A: Project failures

a) Have you ever experienced a software project failure in your organisation? If yes, could you please explain the failure
b) What were the causes of the failure?
c) How did your organisation address the failure?
d) How successful are your software development projects?

Section B: Knowledge management activities

Knowledge need/gap

a) Do you find the need information needs in the organisation?
b) What type of knowledge gap do you usually experience?

Knowledge acquisition

a) Where do you usually find that information that you require?
b) What are the sources of internal and external information?
c) Do you usually find the information from these different sources useful? Please explain.
d) If you don't get it how does it affect the operations of the organisation? Please explain.

Knowledge creation

a) How do you keep your staff up-to-date with information?
b) Do you provide training courses, brainstorming sessions or any activity to update the employees' knowledge and skills?
c) Do you have information resources such as libraries, or maybe books for your employees to update themselves?

Knowledge storage

a) Please explain how information is stored in your organisation?
b) Which devices/ technologies do you usually use to store your information?

Knowledge organisation

a) After storing the information, how do you organise it for easy access and retrieval?

Knowledge transfer

b) Could you explain how knowledge is disseminated in the organisation i.e. internally and externally?
c) Do you hold regular meetings, informal gatherings etc. to share information?

Knowledge application

a) Do you apply this knowledge in your development process? Please explain.
b) Is it effective?

Section C: Post-mortem reviews

a) After each project, do you conduct post-mortem reviews of each project? Please explain.
b) If you don't conduct post mortem reviews, please explain why?
c) After your review, do you compile reports on the reviews?
d) Do you store them? Please explain how.
e) Do you use the stored information for future purposes (in future projects)? If yes, please explain how that has help your organisation.

Section D: Knowledge management benefits and Challenges

a) With the knowledge that you have in the organisation, do you believe that it has benefited and organisation? Please explain.

Knowledge management challenges

a) What challenges do you come across as you manage your knowledge?

Appendix B: Interview schedule for developers

Knowledge acquisition

a) Do you experience an information gap in your task?
b) How do you fill that gap?
c) Which sources do you use to acquire the information you need? Why do you use those sources?

Knowledge creation

a) How do you update yourself with information?
b) If you have an information gap, how do you create your own knowledge to fill that gap?
c) How does the organisation support such activities? Please explain.

Knowledge storage

a) Please explain, how do you store your knowledge?

Knowledge organisation

a) How do you organise your information.
b) Do you have some kind of an information catalogue?

Knowledge sharing

a) Please explain how do you share/disseminate valuable information within or outside the organisation?
b) With whom do you usually share information and why?

Knowledge application

a) Do you use such knowledge in any of your tasks? Please explain.

KM benefits

a) From all this information available has it made you to do things differently from the past? Please explain.

Appendix C: letter seeking authority

01 February 2014

TO WHOM IT MAY CONCERN

RE: Introducing Mr Mzwandile Shongwe – PhD Student at University of KwaZulu Natal

This letter serves to introduce and confirm that Mr Mzwandile Shongwe is a duly registered PhD (Information Studies) candidate at the University of KwaZulu Natal. The title of his PhD research is *'Knowledge Management in Learning Software Organisations in KwaZulu-Natal Province, South Africa".* The purpose of the study is to investigate knowledge management practices in software organisations in the KZN province.

The outcome from the study is expected to improve practice, inform policy and extend theory in this field of study. As part of the requirements for the award of a PhD degree he is expected to undertake original research in an environment and place of his choice. The UKZN ethical compliance regulations require him to provide proof (letter or email) that the relevant authority where the research is to be undertaken has given approval.

We appreciate your support and understanding to grant Mr Mzwandile Shongwe permission to carry out research in your organisation(s). Should you need any further clarification, do not hesitate to contact me or Mr Shongwe.

Thank you in advance for your understanding

Prof Stephen M. Mutula

Dean & Head of School
School of Social Sciences
University of KwaZulu Natal
Private Bag X01 Scottsville, 3209
Pietermaritzburg Campus
Tel: +27 (033) 260 5571
Fax: +27 (033) 260 5092
Cell: +27 712 750 109

Appendix D: Letter seeking authority (by student)

School of Social Sciences
University of KwaZulu Natal
Private Bag X01 Scottsville, 3209
Pietermaritzburg Campus
Tel: +27 (033) 260 5571
Fax: +27 (033) 260 5092

01 February 2014

To whom it may concern

Re: Request to conduct research in your organisation

Dear Sir/Madam,

I am registered PhD (Information Studies) candidate at the University of KwaZulu- Natal. The title of my research is *'Knowledge Management in Learning Software Organisations in KwaZulu-Natal Province, South Africa".* The purpose of the study is to investigate knowledge management practices in software development organisations.

I have identified your organisation as a potential role player in my study. I therefore humbly request authority to conduct the study in your organisation. I intend to interview the software development project leader/manager(s). The interview will take about sixty minutes. I would also like to conduct interviews with only one developer. The interview will take about thirty minutes.

I therefore humbly request authority to conduct my study in your organisation. I absolutely understand time constraints that you might have, but I promise to cooperate with your organisation in every way possible.

I believe that the results of this study will not only be important to me but to your organisation as well. New information that might be useful to your organisation will emerge from this study.

I hope my request will receive your favourable consideration.
My contact details are:

Email:shongwem@unizulu.ac.za (primary) or mzwandileshongwe@yahoo.co.uk (secondary)
Tel: 0359026820
Cell: 073 512 8026
Yours sincerely,
Mzwandile M Shongwe
211550432

Appendix E: Letters granting authority to conduct research

CYBERFOX
innovate create inspire
REG NO: 2005/17310/23 VAT NO: 4190227522
15 Drummond Street, PMB, 3201
Tel: 033 394 2761 / 033 818 9404
Fax: 086 651 7703 Cell: 083 786 9461
E-mail: info@cyberfox.co.za
Web: www.cyberfox.co.za
SOFTWARE DEVELOPMENT : E-LEARNING : TRAINING : WEBSITES
ENGAGE ECM SOFTWARE SYSTEMS
Easy ACADEMIC
LearnQuick
mictseta

28th June 2016

To: Whom it may concern:

Re: Knowledge Management in a Software Organisation:

I hereby confirm that I have granted permission to Mzwandile Muzi Shongwe to conduct research on the above topic within our organisation.

I have also agreed to have our company information published as part of this thesis.

Kind regards

.....
Rakesh Lutchman
CEO

SOFTBYTE COMPUTERS
CK 1998/010733/23

SOFTBYTE COMPUTERS CC

Incorporating TNN Software Inc. Vancouver B.C.
T/A Softbyte Canada Inc.

www.softbyte.co.za
E-mail: contact@softbyte.co.za

P.O. BOX 1158
KLOOF
3640
TEL: 031 464-2289
FAX: 031 464-6393
TONY: 082-4444249
JOHANNESBURG
NICK: 082-4537632
TEL: 011 794-8361

LEADERS IN THE DEVELOPMENT OF SOFTWARE FOR THE
FINANCIAL INVESTMENT, INCOME TAX & LIFE ASSURANCE INDUSTRY

26th June 2016.

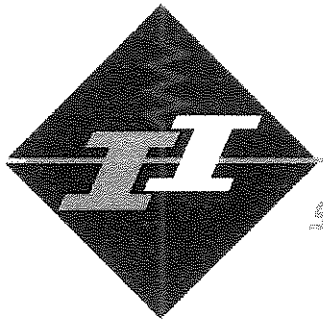
The Ethics Committee
University of KwaZulu Natal
P/Bag X01 Scottsville, 3209

To whom it may concern,

It was my pleasure to engage with Mr Mzwandile M Shongwe on 23rd June 2016 in a discussion regarding the running of certain aspects of the software development side of Softbyte Computers. I feel the discussion was beneficial to both of us.

Yours faithfully,

Anthony Roger de Wijn
CEO Softbyte Computers cc



Inspired Interfaces
Software Developers & Consultants

1 Highland Road
Hillcrest, Durban 3610

PO Box 967 Hillcrest 3650

Telephone: + 27 31 765 6650

Fax: + 27 31 765 6649

Email: info@interfaces.co.za

www.ReticMaster.com

Dean & Head of School
Prof Stephan M. Mutula
School of Social Sciences
University of KZN
socialsciences@ukzn.ac.za

Copy to:
Mr Mzwandile Shongwe
ShongweM@unizulu.ac.za

14 June 2016

Research Approval at Inspired Interfaces

Dear Prof Mutula,

This letter is to confirm that Mr. Mzwandile Shongwe has conducted his research interview at our offices in Hillcrest.

We wish him all the best in this worthwhile research project and that he will achieve great success regarding his doctorate.

Yours Faithfully,

Mr. Francois Nortje
CHAIRMAN

20 December 2016

University of KwaZulu Natal
Private Bag X01 Scottsville, 3209
Pietermaritzburg Campus

To who it may concern

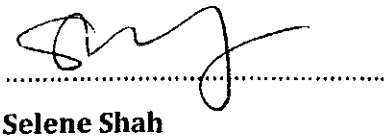
Re: research access granted to Mr Mzwandile Shongwe (211550432)

Dear Sir/Madam

This is to confirm that Mr Mzwandile Shongwe was granted permission to conduct his PhD research in our company, and to interview technical personnel related to his research.

Should you require any further information, please feel free to contact me, Selene Shah.

Yours truly,



Selene Shah
Talent, Learning & Culture
immedia

031 566 8000 • selene@immedia.co.za

Cosoft
Coherent Software Solutions

18 York Road
Gillitts
3610

15 June 2016

The Ethics Committee
University of KwaZulu Natal
Private Bag X01 Scottsville, 3209
Pietermaritzburg Campus

Access granted to Mr Mzwandile Shongwe to conduct research in our company

Dear Sir/Madam,

I would like to confirm that Mr Mzwandile Shongwe was granted access to our company Cosoft to collect data (Interviews) for his PhD research.

Should you need more information, please don't hesitate to contact me using details given above/below.

Yours truly,

Bernd von Fintel

Product Manager
083 701 8583
bernd@cosoft.co.za

ComputASSIST
Pharmacy & Retail Solutions

Dedicated to providing pharmacy with simple effective solutions

18 York Road,Gillitts
Durban
3640
Tel: 031 767 0796
Fax: 031 767 0590

27 June 2016

The Ethics Committee
University of KwaZulu-Natal
P/B X01, Scottsville
Pietermaritzburg
3209

To whom it may concern

Re: research access granted Mr Mzwandile Shongwe to conduct research

Please note that Mr Mzwandile Shongwe was granted access to our organisation to conduct research for his PhD research.
Should you need further information, please do not hesitate to contact me on the address given above.

Yours faithfully,

Rhain Stander
Selaelo Tlabela

Mzwandile Muzi Shongwe

From: Alan Hastings [mailto:alanh@houndsoft.com]
Sent: 31 October 2014 09:32 AM
To: Mzwandile Muzi Shongwe
Subject: FW: Interview for Your Thesis

Is this perhaps what you were looking for?

Kind regards

Alan Hastings
031 764 15 15
083 326 0894
alanh@houndsoft.com

From: Alan Hastings [mailto:alanh@houndsoft.com]
Sent: 22 September 2014 12:49
To: shongwem@unizulu.ac.za
Cc: 'socialsciences@ukzn.ac.za'
Subject: Interview for Your Thesis

Dear Mr. Shongwe

I hereby confirm that we will participate in your interview as agreed.

Kind regards

Alan Hastings
031 764 15 15
083 326 0894
alanh@houndsoft.com

Mzwandile Muzi Shongwe

From: Howard Hudson [britnat@gmail.com]
Sent: 08 October 2014 09:07 AM
To: Mzwandile Muzi Shongwe
Subject: Confirmation of Appointment

From:

Bridge Business Solutions

42 Bridge Road,

Zwartkops,

Pietermaritzburg

Attention: Mzwandile Shongwe.

Good morning Mzwandile,

Thank you for your interest, this serves to confirm that we have agreed to meet with you to answer questions related to your research project. (Companies active in I.T. technology / software development)

Yours sincerely

HF Hudson

For Bridge Business Solutions

072 832 6445

www.bridgebizsolutions.co.za

Mzwandile Muzi Shongwe

From: Bob Ludlow [mailto:Bob@albertinabay.com]
Sent: 30 October 2014 04:20 PM
To: Mzwandile Muzi Shongwe
Subject: Knowledge Management in learning software organisations in KwaZulu-Natal Province

Dear Mzwandile,

It a pleasure for our organisation to assist you in your research into "Knowledge Management in learning software organisations in KwaZulu-Natal Province".

Bob Ludlow
Albertina Bay Technologies cc

+27 31 764 8556 (Tel Direct)
+27 82 653 6501 (Cel)
+27 31 764 8550 (Company)
+27 31 764 8568 (Fax Company)
http://www.albertinabay.com

Mzwandile Muzi Shongwe

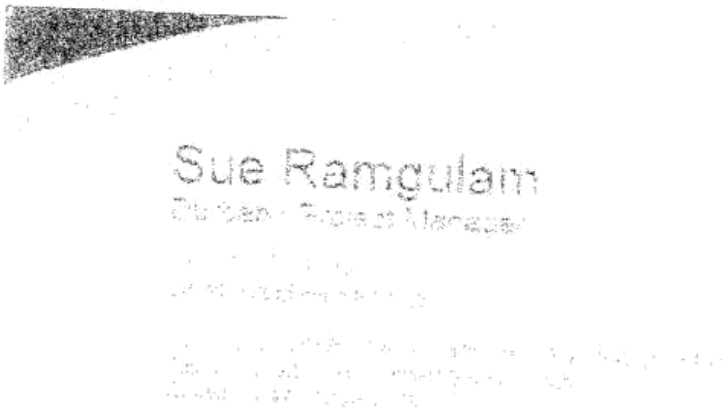
From: Sue Ramgulam [sue@implyit.co.za]
Sent: 30 October 2014 11:40 AM
To: Mzwandile Muzi Shongwe
Cc: Bruce Pieterse
Subject: Study Conducted

Good Day,

I would like to confirm that we have participated in Mr Shongwe's study.
Should you have any queries, please feel free to contact me.

Thank you.

Kind Regards,



ACCPICK XCELLENCE

13 January 2015

RE: Knowlegde Management Study

To UKZN Ethics Committee,

We have been asked to participate in a study and provide information about how our organisation manages its' knowledge from both internal and external sources, in the providing and supporting of our software products.

We are happy to participate in the survey/interview questions provided to us to the best of our knowledge.

Regards,

Ryan Pretorius
Manager

Independent Systems

Members: F.P. Maree

CK # 1988/019605/23 Vat # 4960101998 Email: pos@accpick.co.za P O Box 1187 Hilton 3245 **Tel: +27 33 343 6000 Fax: +27 33 343 6001**

Phone: 033 343 2888 ext 105
Fax: 033 343 1500
Email: john@summitsolutions.co.za
Website: http://www.summitsolutions.co.za

For attention: The UKZN Ethics Committee

This Letter serves to inform you that Mzwandile Shongwe has been authorised to gain access to our organisation and given him permission to utilise information provided by us for the purposes of his research.

Kind Regards

John Kiln

Appendix F: Informed consent document

Dear Participant,

My name is Mzwandile Muzi Shongwe I am a PhD (Library and Information Science) candidate studying at the University of KwaZulu-Natal, Pietermaritzburg Campus. The title of the research is: Knowledge Management in Learning Software Organisations in KwaZulu-Natal Province, South Africa. The aim of the study is to investigate whether software development organisations have adopted knowledge management and the benefits of knowledge management in the organisation.

I am interested in interviewing you so as to share your experiences and observations on the subject matter.

Please note that:

- The information that you provide will be used for scholarly research only.
- Your participation is entirely voluntary. You have a choice to participate, not to participate or stop participating in the research. You will not be penalized for taking such an action.
- Your views in this interview will be presented anonymously. Neither your name nor identity will be disclosed in any form in the study.
- The interview will take about one hour.
- The record as well as other items associated with the interview will be held in a password-protected file accessible only to myself and my supervisors. After a period of 5 years, in line with the rules of the university, it will be disposed by shredding and burning.
- If you agree to participate please sign the declaration attached to this statement (a separate sheet will be provided for signatures)

I can be contacted at: School of Social Sciences, University of KwaZulu-Natal, Pietermaritzburg Campus, Scottsville, Pietermaritzburg.
Contact Details:
Email: shongwem@unizulu.ac.za
or mzwandileshongwe@yahoo.co.uk;
Cell: 0735128026
My supervisor is Professor Steven Mutula who is located at the School of Social Sciences, Pietermaritzburg Campus of the University of KwaZulu-Natal.
Contact details:
Email: mutulas@uknz.ac.za;
Phone number: 0332605571

The College of Humanities Research Ethics Officer is Phumelele Ximba who is located at Humanities Research Ethics Office, University of KwaZulu-Natal. Contact details: email: ximbap@ukzn.ac.za Phone number +27312603587.

Thank you for your contribution to this research.

DECLARATION

I……(full names of participant) hereby confirm that I understand the contents of this document and the nature of the research project, and I consent to participating in the research project. I hereby consent / do not consent to have this interview recorded.

I understand that I am at liberty to withdraw from the project at any time, should I so desire. I understand the intention of the research. I hereby agree to participate.

SIGNATURE OF PARTICIPANT DATE

……

UNIVERSITY OF KWAZULU-NATAL
INYUVESI YAKWAZULU-NATALI

Appendix G: Ethical clearance certificate

11 January 2017

Mr Mzwandile Muzi Shongwe (211550432)
School of Social Sciences
Pietermaritzburg Campus

Dear Mr Shongwe

Protocol reference number: HSS/0724/014D
Project title: Knowledge Management in Learning Software Organisations in KwaZulu-Natal Province, South Africa

Full Approval – Expedited Application

In response to your application received 3 July 2014, the Humanities & Social Sciences Research Ethics Committee has considered the abovementioned application and the protocol has been granted **FULL APPROVAL.**

Any alteration/s to the approved research protocol i.e. Questionnaire/Interview Schedule, Informed Consent Form, Title of the Project, Location of the Study, Research Approach and Methods must be reviewed and approved through the amendment /modification prior to its implementation. In case you have further queries, please quote the above reference number.

PLEASE NOTE: Research data should be securely stored in the discipline/department for a period of 5 years.

The ethical clearance certificate is only valid for a period of 3 years from the date of issue. Thereafter Recertification must be applied for on an annual basis.

I take this opportunity of wishing you everything of the best with your study.

Yours faithfully

.....

Dr Shenuka Singh (Chair)
Humanitities & Social Scinces Research Ethics Committee

/pm

Cc Supervisors: Professor Stephen M Mutula
Cc Academic Leader Research: Professor Maheshvari Naidu
Cc School Administrator: Ms Nancy Mudau