The success rate of software projects in South African businesses

RT Coetzee

orcid.org 0000-0003-0578-4572

Mini-dissertation submitted in partial fulfilment of the requirements for the degree *Master of Business Administration* at the North-West University

Supervisor: Prof GJ de Klerk

Co-supervisor: Prof CJ Botha

Graduation ceremony: October 2018

Student number: 26926105

Abstract

This study will analyse and investigate software projects and their success rate in South African businesses, which include the private and public sectors. A survey will be developed and distributed among different people within companies, who occupy roles ranging from project managers and stakeholders to users and developers. This, together with a literature review, will produce a clear picture of the success and failure rates of projects and the likely causes of the failures and successes in South African businesses.

Keywords: Software project failure, IT, Causes of software project failures, Software project success, ICT, Software project management, Agile, Waterfall, Development methodologies.

Abbreviations and definitions

Agile: A software development method of dividing tasks into short phases of work, which includes frequent reassessment and adaptation of plans.

Analyst: An individual whose job it is to analyse a system.

Cost-benefit analysis: An approach for estimating the strengths and weaknesses of alternatives.

Deliverable: A tangible or intangible good or service produced as a result of a project.

Developer: An individual who develops a software system.

ERP: Enterprise resource planning.

ICT: Information and communication technology – an extended term for IT that encompasses all technology.

Impediment: An obstruction in a project that is stopping it from moving forward.

IT: Information technology.

Product owners: The individual who is responsible for what the business wants.

Project steering committee: The group that is responsible for the business issues associated with a project.

Scope: The functionality the system will cover in the current project.

SDLC: Software development life cycle – planning, analysis, design, implementation, and support.

Software project: Any project that has as its end goal the delivering of a technology to the user.

Sponsor: The individual with overall accountability for the project.

Tester: An individual who tests a software system to make sure that it performs the way it should.

Time to value: The term that describes the period between a request for a deliverable and the initial delivery.

Users: The users of the system, the people who are going to work with the system on a day-to-day basis.

Waterfall: A software development method of sequential stages and a fixed plan of work.

Table of contents

List of tables

List of figures

Chapter 1: Nature and scope of the study

1.1. Introduction

In today's contemporary software development environment, companies are faced with ever-changing business requirements as well as technological changes that need to be weighed up against alternatives (Baron & Spulber, 2017). The fast-changing world is causing uncertainty and concern for businesses (Farrow, 1997). The use of technology in production has changed the way organisations compete, with new technology continually being applied in response to market transformations and trends. This is affecting whole industries and markets. "The ability to adapt in a rapidly changing and complex environment has thus become an increasingly important aspect of competitiveness" (Sonntag, 2003). Software projects are getting larger and are pervading all departments of organisations, which can pose a risk for the organisation should something go wrong (Bloch *et al.*, 2012).

One of the most concerning aspects of software development is the high failure rate of projects. According to a study done by the Gartner Group (2015), only 32% of all projects were successful, delivering what was promised, on budget, and on time. Forty-four per cent of projects were late, over budget, and/or delivered less functionality than was originally in the scope. Twenty-four per cent failed and were cancelled before completion (Ezer, 2010). Some information technology (IT) projects with budgets that exceeded $15 million ran late, while 45% were over budget and delivered less value than originally planned (Bloch *et al.*, 2012).

Software development has changed over the last couple of years, with new methodologies being developed to address the shortcomings of more traditional methodologies. Compared to building a skyscraper, software development differs with respect to how fast things are changing. When building a skyscraper, there is not much flexibility in changing the design while the building is being built. All the designs are worked out before construction starts and from that point onwards no changes can be made to the specification and blueprints, which makes it easier to control the budget and the timelines because of the amount of time that was spent on the design phase. On the other side of the spectrum is software development, which is engaged in

developing a system that can't be seen with the eyes, as opposed to a building where progress can be physically seen. Business requirements change faster than the design of a software system happens, thus making a static design like a building not feasible. This difference makes it harder to control budgets and timelines, and continuous change is one of the reasons for the creation of new methodologies that embrace change.

The literature frequently reports that software projects fail and are abandoned before completion. Stakeholders often see software project development as underperforming, failing to meet expectations, and not delivering value to customers. Due to these shortcomings project management continues to grow and play a more important role in the delivery of software projects. It has become a multi-disciplinary field with distinctive tools and methodologies.

A software methodology can be described as a framework that is used to structure, plan, and control the process of developing an information system (IT Knowledge Portal, 2018). Over the last few decades, more software delivery methodologies have been created than ever before. Figure 1.1 contains a timeline from 1968, which shows how many different methodologies have evolved over the last five decades.

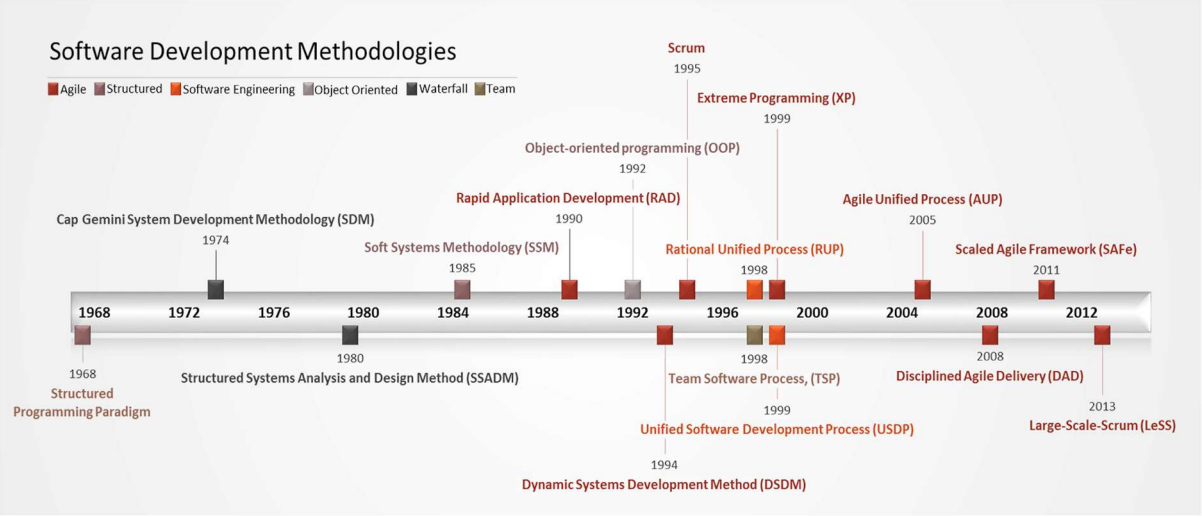

Figure 1.1 Software development methodologies timeline (Anon, 2016)

All of these methodologies strive to address the software development issues that businesses face on a day-to-day basis. Survey responses from IT professionals and

business users indicate that both groups share concerns regarding software development projects. The main findings include (Geneca, 2017):

- 75% of respondents feel that their projects were doomed right from the start.
- 80% admitted that they spend at least 50% of their time on rework.
- 78% felt that stakeholders need to be more involved in the requirement process.
- Only 55% felt that the objectives of the project were clear to them.

Other issues might include how best to utilise the software development team which is comprised of many types of expertise, how to track progress, how to plan better, and how to plan budgets and track costs.

The following methodologies are the main ones in use today:

- Crystal
- Dynamic System Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FFD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- System Development Life Cycle (SDLC)
- Waterfall/Traditional

When it comes to project failures, it is difficult to calculate the cost of failed software projects (Jørgensen *et al.*, 2017). If a decision made early in a project sets the course for failure, the cost of fixing that mistake later in the project is exponentially higher compared to an earlier fix. The cost of fixing errors later in the software development cycle increases exponentially because the artefacts build on each other (Ambler, 2010).

The high cost of software project failure warrants research in an attempt to minimise these financial losses and to determine what the causes of these failures might be. When the reasons for failure are better understood and identified, South African businesses can put corrective actions and new methodologies in place to try to minimise the financial losses of potential project failures.

1.2. Problem statement

Over the last couple of years, research has been done to identify the reasons why software projects fail. However, this research does not always provide enough detail to understand the full picture of project failure and the reasons behind this failure (Verner *et al.*, 2008). Projects do not fail for one single reason; there is usually more than one reason why projects fail (Anon, 2018). Understanding these reasons and the way they interact with delivery methodologies of software projects can create a clearer picture of what to avoid and what to do when faced with these issues.

Software projects fail at a high rate across the globe, costing companies billions of dollars each year. Many reasons have been presented for this failure rate. The question arises whether this happens in South Africa as well. If so:

- What are the causes of these failures internationally and in South Africa?
- Are there methodologies that have higher success rates?
- Can identifying these challenges and methodologies increase the possibility of better success rates?

Combining the knowledge of why projects fail with the advantages and disadvantages of different methodologies could potentially lead to better success rates of projects.

1.3. Objectives of the study

1.3.1. Primary objective

The primary objective of this study is to explore the South African business environment to identify software project success/failure rates and potential causes for these successes and failures.

1.3.2. Secondary objective

The secondary objective of this study is to determine whether there are methodologies that could be used by South African businesses which could help increase the success rate of software projects within businesses.

1.4. Scope of the study

The scope of the study will include any software development project within South Africa with the end goal of delivering value to a customer in the South African business environment.

1.5. Research methodology

The aim of this research is to determine if software projects fail in South Africa, and what some of the reasons for these failures may be. Current literature was consulted, and these findings were used as a basis for compiling research questionnaires. Descriptive statistics were used as the base of the research method.

Table 1.1 explains the primary research questions, the methods used to answer the questions and the expected outcomes.

Research method		
Objective	Method	Expected outcome
Primary objective	Literature study	**Chapter 2:** Literature review
Secondary objective	Literature study and survey questionnaire	**Chapter 3:** Empirical study **Chapter 4:** Conclusion and recommendations

Table 1.1 Research method

1.5.1. Literature/theoretical study

Chapter 2 includes a complete review of software projects. The sources consulted for secondary data are listed below.

- The internet
- Science journals (*Institute of Electrical and Electronics Engineers and Journal of Information Technology*)
- Management journals (*Journal of Management* and *Academy of Management*)
- Magazines (*Stuff*, *Smart Computing*, and *Technology Review*)
- Institutions (Statistics SA, companies operating in the information technology space)
- Books
- Papers relevant to the subject
- Financial reports (integrated reports, newspapers)

Keywords: Software Projects; ICT; SDLC; Deliverables; Project fails; Requirements; Late delivery; Agile; Waterfall; Cost benefit analysis; Project plan; time to value; change management.

1.5.2. Empirical study

The primary data collection in Chapter 3 was based on online surveys and emailed survey questionnaires, which have proven to be the most practical method of data collection. The secondary data was gathered from data available in reports.

1.5.3. Data analysis and techniques

Statistical methods were used to analyse the quantitative data received from the survey. This data is presented in graph and table formats.

1.6. Limitations of the study

There are several limitations to consider when interpreting this study's outcomes. These limitations include:

- That software projects have not been studied extensively in South Africa.
- That the two major sources of data, Gartner and Standish, focus predominantly on American studies at a cost of R16 280 to R39 000 per report. Data available in summary format was used.
- That the sample size in South Africa is relatively small.
- That companies are not open about project failures. Data regarding failures is not always accessible in the public domain.
- That employees will rarely take responsibility for project failures.

1.7. Layout of the study

Chapter 1: Nature and scope of the study

The focus in Chapter 1 is on outlining the reason for the study, and to understand the significance of studying software projects as well as the possible causes of their failures.

Chapter 2: Literature review

The focus in Chapter 2 is to research and present the reasons why software projects fail. The South African environment will be investigated to shed some light on the type

of environment and some of its challenges. Software project failure statistics will be listed, with a list of reasons for these failures. A detailed investigation into the 10 main reasons for software project failure as identified by the Standish Group will be discussed. The chapter will conclude with an analysis of different methodologies and what benefits they have.

Chapter 3: Empirical study

The focus of Chapter 3 is to discuss and analyse the findings of the survey responses. The key findings of what the success rate of software projects in South African business will be discussed in detail, together with the main reasons for failure and factors that will lead to better success.

Chapter 4: Conclusions and recommendations

The focus of Chapter 4 is to draw conclusions based on the literature review and empirical study. In addition, proposals for future studies will be made.

Chapter 2: Literature review

2.1. Introduction

The business environment has changed over the last few decades with technology pushing the boundaries and opening the world to a truly global market. Never before has IT played as an important a role as it does now. In South Africa, “the direct contribution of the ICT (information and communication technology) sector to the gross domestic product (GDP) was R94,7 billion (or 2,9% of total GDP) in 2012. Compared to other industries, the ICT sector contributed more to South Africa’s economy than agriculture” (Statistics SA, 2017). This shows just how fast technology is becoming part of everyday life.

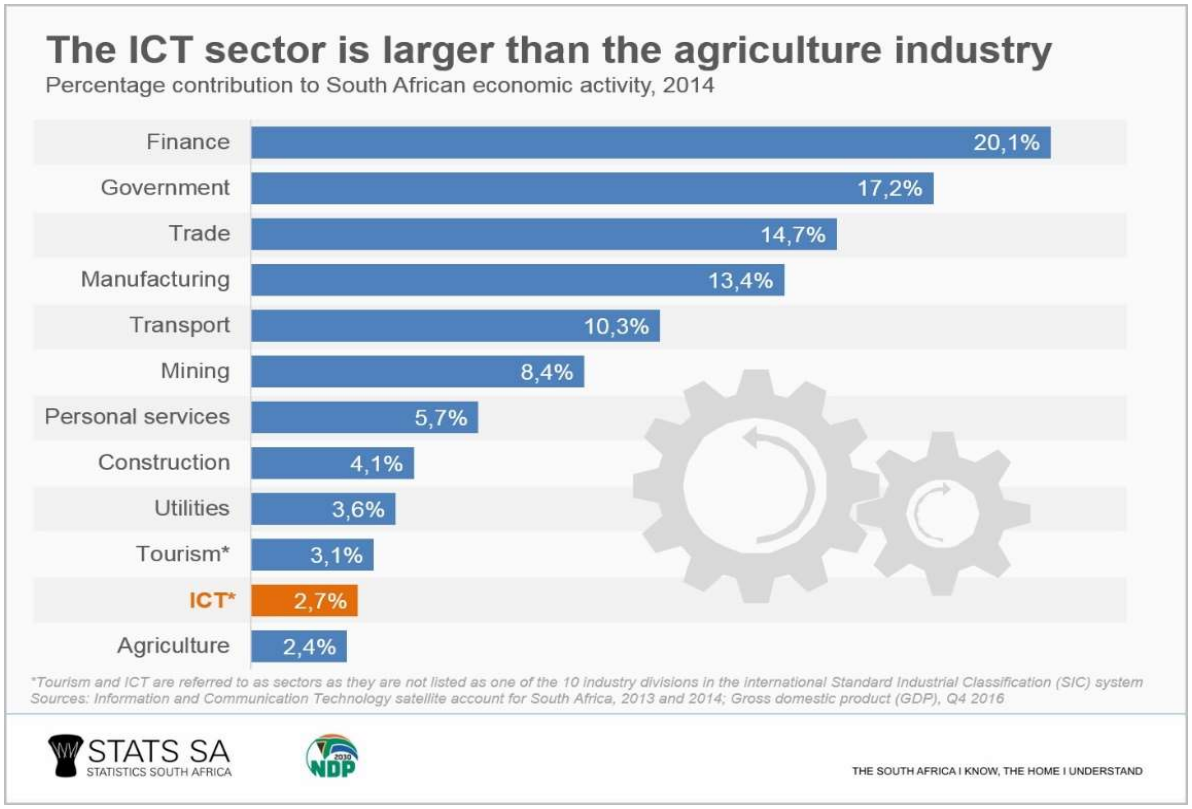

Figure 2.1 ICT sector contribution (Statistics SA, 2017)

The more technology becomes part of life, the more it starts to become important for organisations. Historically, IT departments were perceived as cost centres and were detached from the rest of the organisation. In more recent times, companies such as Walmart have combined their corporate IT and e-commerce technology groups, thus

moving all IT to the heart of business growth (Nash & Norton, 2016). Despite of this move towards technology and the importance of it in business and government, it is still a well-known fact that software projects fail. In the United States of America, 31,1% of projects will be cancelled before they are completed (Standish Group, 2014). Further results show that 52,7% of projects will cost 189% of the original estimate (Standish Group, 2014). It is important for South African businesses to be aware of these trends. Because South African businesses are generally consumers of international technology, they need to know what the pitfalls are and what methods tend to lead to more positive project outcomes. The previous chapter provided an introduction and background to the study. The purpose of this chapter is to present, deliberate on, and study software projects in South Africa and the international arena in order to determine why projects fail and what methodologies can be used to increase projects' success rates.

2.2. The South African landscape

2.2.1. The economy

Economic growth of 1,3% is projected for South Africa in 2017 (National Treasury, 2017), but 1,3% is not enough to reduce the inequality that exists in the country and to create more jobs for the unemployed. Even with this low growth rate, the South African formal business sector is generating higher amounts of income every year, but costs have escalated at a slightly faster rate (Figure 2.2). As a result, profit has declined in relative terms (i.e. as a percentage of income) (Statistics SA, 2016).

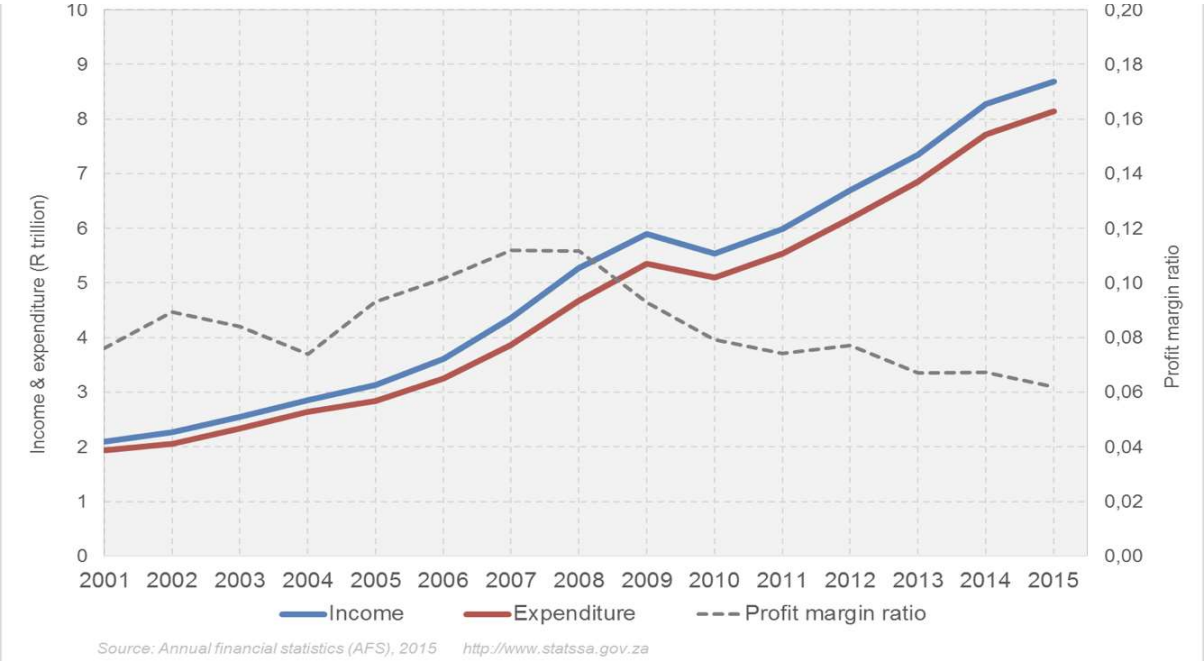

Figure 2.2 Income, expenditure and net profit margin (Statistics SA, 2016)

With businesses making less profit due to higher expenses, it is becoming increasingly important to use technology and innovation to drive success. “Innovation is what steam was in the industrial revolution” (BusinessVibes, 2015). For South Africa to stimulate its economic growth and improve its competitiveness, it needs to rely on science and technology to a greater degree (Department of Science and Technology, 2007). One of the McKinsey Group’s priorities for inclusive growth is advanced manufacturing, where South Africa needs to draw on its skilled labour to grow into a globally competitive manufacturing hub. To do this, South Africa will have to pursue new markets and step up innovation and productivity (Leke *et al.*, 2015).

2.2.2. The information and communication technology environment

In South Africa, there is a particularly turbulent ICT environment, with the public sector investing huge amounts of money. These projects do not always go as planned, with some of the biggest scandals and failed software projects having become known. These projects are an important part of South Africa’s competitive advantage and need to be implemented for South Africa to match other technology leaders. If South Africa does not keep up with current technology and implement the latest software in all sectors of business, it will soon be lagging in this area.

According to a study performed at Transnet, “The management dilemma is that Transnet has committed R65 billion to projects in the hope of developing its core businesses to that of world-class standards as a logistics service provider in South Africa. Transnet’s capital project division, Protekon, is by and large responsible for managing the projects committed to this R65 billion capital expenditure. However, Transnet’s perception of Protekon’s project management abilities is very poor. This lack of confidence and a perception of project failure often retards the project selection and award, the commitment of funds, and by extension Transnet’s commitment of delivery to its customers” (Pillay, 2006).

One of the biggest software blunders in recent years was the Integrated Financial Management System at the South African National Treasury. “Former finance minister Pravin Gordhan, his former deputy Mcebisi Jonas and former Treasury director-general Lungisa Fuzile are under fire over a failed R1 billion IT project” (Mkentane, 2017). The National Treasury initiated the Integrated Financial Management System in 2005 and spent R1 billion and R1,2 billion on the first and second phases respectively. There are complaints that the National Treasury did not get value for money from its IT systems in the past decade (Mkentane, 2017). The system that was supposed to integrate all human resources and financial management systems across government has yet to add any value.

Another software project that caused frustration for residents was a project conducted by the city council of Johannesburg. Residents received inflated bills, and many had their services cut off. These mishaps led to the implementation of a new IT project known as Programme Phakama, which was designed to improve the city’s billing system (Anon, 2011).

In the private sector, failures are not made as public as they are in the public sector. Companies rarely publicly admit to software failure. Edcon, in their 2016 annual integrated report, mentioned that if their IT and telecommunication systems experience interruptions, it could lead to a deterioration of ABSA’s ability to process credit applications and collect payments. This would lead to reputational damage and result in loss of business (Edcon Holdings Limited, 2016).

Smith (2012:2) states that the main reason why projects fail in South Africa is changing requirements. He goes on to state that, for the most part, South African companies and international companies share the same reasons for software project failures. The top three reasons are incomplete requirements and specifications, changing requirements and specifications, and lack of user input.

2.3. King reports

The King III Report (Institute of Directors, 2009) has the following to say about information technology: “Information systems were used as enablers to business, but have now become pervasive in the sense that they are built into the strategy of the business. The pervasiveness of IT in business today mandates the governance of IT as a corporate imperative.” The report goes on to elaborate on how important IT has become: “In most companies, IT has become an integral part of the business and is fundamental to support, sustain, and grow the business. Not only is IT an operational enabler for a company but is also an important strategic asset to create opportunities and to gain a competitive advantage. Companies have made, and continue to make, a significant investment in IT. Virtually all components, aspects, and processes of a company include some form of automation. This has resulted in companies relying enormously on IT systems. Further, the emergence and evolution of the internet, e-commerce, online trading, and electronic communication have also enabled companies to conduct business electronically and perform transactions instantly. These developments bring about significant risks and should be well governed and controlled. We, therefore, deal with IT governance in detail in King III for the first time” (Institute of Directors, 2009).

The risk of IT systems is ever growing. “There is no doubt that the complexity of IT systems does create operational risks and when one outsources IT services, for instance, this has the potential to increase risk because confidential information is outside the company. Consideration has to be given to the integrity and availability of the functioning of the system; possession of the system; authenticity of system information; and assurance that the system is usable and useful. Concerns include unauthorised use, access, disclosure, disruption, or changes to the information system” (Institute of Directors, 2009).

The King III report lists the following seven principles for the governance of IT (Institute of Directors, 2009:82-87):

1. The board should be responsible for IT governance.
2. IT should be aligned with the performance and sustainability objectives of the company.
3. The board should delegate to management the responsibility for the implementation of an IT governance framework.
4. The board should monitor and evaluate significant IT investments and expenditure.
5. IT should form an integral part of the company's risk management.
6. The board should ensure that information assets are managed effectively.
7. A risk committee and audit committee should assist the board in carrying out its IT responsibilities.

The King IV report echoes King III seven years after the latter was released (Institute of Directors, 2016): "Technology governance and security have become critical issues. Technology is no longer simply an enabler; the systems created by an organisation provide the platform on which it does business, and technology is now both the source of many of an organisation's future opportunities and potential disruption – an excellent example of how risk and opportunity are increasingly two sides of the same coin. Technology is now part of the corporate DNA. Thus, the security of information systems has become critical. Technology governance and security should become another recurring item on the governing body's agenda."

Technology and information have advanced and revolutionised business and societies and transformed products, services, and business models to the point where many believe they have heralded the dawn of a Fourth Industrial Revolution (Institute of Directors, 2016:30). The King IV report (Institute of Directors, 2016:62-63) has as its twelfth principal: "The governing body should govern technology and information in a way that supports the organisation setting and achieving its strategic objectives", and lists the following recommended practices:

- The governing body should assume responsibility for the governance of technology and information by setting the direction for how technology and information should be approached and addressed in the organisation.
- The governing body should approve a policy that articulates and gives effect to its set direction on the employment of technology and information.
- The governing body should delegate to management the responsibility to implement and execute effective technology and information management.
- The governing body should exercise ongoing oversight of technology and information management and, in particular, oversee that it results in the following:
    - Integration of people, technologies, information, and process across the organisation.
    - Integration of technology and information risk into organisation-wide risk management.
    - Arrangement to provide for business resilience.
    - Proactive monitoring of intelligence to identify and report incidents, including cyberattacks and adverse social media events.
    - Management of the performance of, and the risk pertaining to, third-party and outsourced service providers.
    - The assessment of value delivered to the organisation through significant investment in technology and information, including the evaluation of projects throughout their life cycles and of significant operational expenditure.
    - The responsible disposal of obsolete technology and information in a way that has regard to environmental impact and information security.
    - Ethical and responsible use of technology and information.
    - Compliance with relevant laws.
- The governing body should exercise ongoing oversight of the management of information and, in particular, oversee that it results in the following:
    - The leveraging of information to sustain and enhance the organisation’s intellectual capital.
    - An information architecture that supports confidentiality, integrity, and availability of information.

    - The protection of privacy of personal information.
    - The continual monitoring of security of information.
- The governing body should exercise ongoing oversight of the management of information and, in particular, oversee that it results in the following:
    - A technology architecture that enables the achievement of strategic and operational objectives.
    - The management of the risk pertaining to the sourcing of technology.
    - Monitoring and appropriate responses to development in technology, including the capturing of potential opportunities and the management of disruptive effects in the organisation and its business model.
- The governing body should consider the need to receive periodic independent assurance on the effectiveness of the organisation's technology and information arrangements, including outsourced services.
- The following should be disclosed in relation to technology and information:
    - An overview of the arrangements of governing and managing technology and information.
    - Key areas of focus during the reporting period, including objectives, significant changes in policy, significant acquisitions, and remedial actions taken as a result of major incidents.
    - Actions taken to monitor the effectiveness of technology and information management and how the outcomes were addressed. Planned areas of future focus.

2.4. What is software project success?

Agarwal and Rathod (2006) mention that successful projects are rare, and that one reason for this may have to do with different perceptions of success in the minds of those evaluating a project. A person external to the project might use cost and return on investment to determine the success of the project. However, someone internal to the project might look at the deliverables and functionality of the project compared to the agreed scope of the project.

Several decades ago, Pinto and Slevin (1988) stated that the concept of project success has been ambiguously defined, indicating that projects that finished late and

were over budget were seen as successes. Yet other projects that were on time and within budget limits were considered failures. The definition of a successful project, according to Sam Bad (2014), is based on the project management body of knowledge and has the following nine criteria:

Time frame: Successful projects should take place as close as possible to the baseline plan. Any deviation or variation is unpleasant. One can measure schedule success as a percentage: 1 - (actual time - planned time)/planned time * 100 = %.Cost: Successful projects should take place as close as possible to the planned budget. The cost success can be used as a percentage: 1 - (actual cost - planned cost)/planned cost * 100 = %.

Communication: Successful projects should take place with the least conflict, most collaboration, and ease of communication. This could have subjective measures, such as a project team's rating of ease of communication. There can also be objective measures, such as the number of conflicts that arose as a percentage of the number of conflicts resolved.

Scope: Successful projects should take place with the least unnecessary changes and the most effective configuration management. One can use the following ratio to measure scope success: (number of reasonable changes completed/number of reasonable change requests) %.

Stakeholder: This is subjective, but a successful project should leave the stakeholders satisfied and willing to do further projects.

Quality: Quality standards, such as Statistical Quality Control, can be used to measure the quality of outcomes.

Human resources: Successful projects should use the least human resources and keep the team members involved and satisfied.

Procurement: Successful projects should enhance relationships with vendors and other members of the supply chain.

Risk: Successful projects should take place with minimum risk. Risk can be measured by the number of risky events that were identified and how each risk was dealt with.

It is clear that there have been many definitions for project success over the last couple of years. The Standish Group (2014) breaks projects into three categories that are generally considered a baseline for classifying projects.

1. Project success: A project that is completed within the agreed-upon time frame and budget, with all features and functions as initially specified.
2. Project challenged: A project that is completed and operational but is over the agreed-upon time frame and budget, and delivering fewer features and functions than originally specified.
3. Project impaired: A project that is cancelled altogether during the development cycle.

The focus of this study will be on failed projects and the reasons for their failure. Only when the causes of project failure are fully understood can one implement systems that would prevent such failures from being repeated in the future.

2.5. Project statistics

Standish Group reports will be used as a benchmark for project statistics and reasons for project failures. (For the sake of the simplicity of the study both challenged and failed projects will be grouped under failed.) Between the years 2011 and 2015 there was no significant deviation in project success rates. Data for those five years shows a steady project success rate, as seen in Table 2.1.

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Table 2.1 Modern resolution for all projects (Standish Group, 2015)

The size of a project seems to play an important role in project success or failure. Small projects showed a success rate of 62% and a failure rate of only 11%. In contrast, reports for grand projects showed only 2% to have been successful, as shown in Table 2.2.

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

Table 2.2 Chaos resolution by project size (Standish Group, 2015)

The success rate between Waterfall and Agile projects is also interesting to note, with Agile projects having a higher success rate of 42% compared to Waterfall at 14% (Standish Group, 2015), as shown in Figure 2.3.

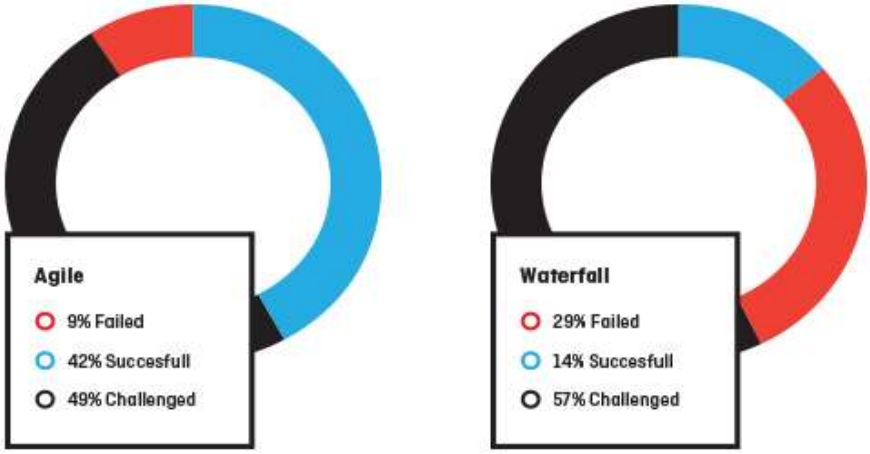

Figure 2.3 Chaos report by methodology (Standish Group, 2015)

The findings presented up to this point clearly indicate that projects fail at higher rates than they should. The following is an investigation into the main causes of software project failures.

2.6. Main causes of software project failures

In this section, an analysis of factors that might lead to project failures will be done. Again, Standish Group reports will be used as a starting point for understanding root causes of failed projects.

The Standish Group lists 10 main causes of failed projects (Figure 2.4). These 10 causes will be explained and researched in detail to form a better understanding of their impact on South African businesses.

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

Figure 2.4 Project challenged factors (Standish Group, 2014)

2.6.1. Lack of user input

According to the Standish Group's list of factors that lead to project failures, the lack of user input ranks as the number one reason in 12.8% of cases. A new system may change the way people work. Designing a new system requires intimate knowledge of how a group of people work, in order to ensure that the proposed system supports them well (Beyer & Holtzblatt, 1995). User involvement is playing an increasingly important role in system development. Meaningful user involvement in the system development cycle is critical to the success of any project (Sun, 2013).

When users are involved in system development, it increases their willingness to use the new system. It also makes for a better design and a more positive attitude towards the system in general. The earlier on in the system development life cycle the users are involved, the more understanding they will provide to the system developers as to what needs to be developed. When it comes to the initial design of the system, the existing problems need to be communicated to the system designers. If the users of the system are omitted at this stage, the problem and the requirements might be misinterpreted and can lead to designing a system that will not solve the business problems.

Because designing a new system is so closely linked to how people do their work, designers need effective methods of gathering information (Beyer & Holtzblatt, 1995). If the users are not consulted with this understanding in mind, the users might not be willing to interact with system developers and the true nature of the problem might not be fully understood. If this misunderstanding of user requirements is picked up too late in the development process, it will cost exponentially more to fix than making sure the users are involved from the start.

Thus, including system users as early as possible will ensure that the correct problem is identified and addressed during the design of the system. Users will be more willing to get involved and in turn be more willing to adopt the new system once it is completed.

2.6.2. Incomplete requirements and specifications

The development of software requirements is done by collecting information about the anticipated system in a progressive manner. This process of gathering information can be supported by an analysis-revision cycle, by which the analysis phase checks the correctness of the information, and in the revision phase corrects any problems once they are detected (García-Duque *et al.*, 2009).

Recently, models and techniques have been put forward in an effort to streamline the analysis phase of the software development life cycle. Some of the popular models are: Waterfall, Incremental, Spiral, and Agile. All of these models have requirement gathering as the first step in the process, indicating just how important the requirements and specifications are to the process. These models are of the “utmost importance for delivering the software in a systematic manner such that it will be delivered within deadline and should also have proper quality” (Modi *et al.*, 2017). A more detailed analysis of the various methodologies will be done in Chapter 2.7.

Should the requirement and specification of the system be incomplete or ill-defined, it will lead to building solutions that will in fact not solve the problem it was intended to solve. It is thus of utmost importance to define the correct requirements and specifications from the perspective of the relevant stakeholders before the software is built.

2.6.3. Changing requirements and specifications

Changes to the requirements happen when, during the project, there is a departure from the original requirements as stipulated in the analysis phase at the start of the project. “Change in software requirements during its development phase is considered as a major risk which may affect a software project to fail” (Aljohani & Qureshi, 2016). Perhaps the most noticeable reason for requirements to change is that it was not possible to capture them properly to start off with (Kelly, 2004). Another reason for change are variations in circumstances. This can include external changes, internal changes, and technical changes (Kelly, 2004).

When looking at the Standish Group report, changing requirements are third on the list of reasons for project failure, after lack of user input and incomplete requirements. If these two factors are omitted, and the user was not consulted from the start and the requirement gathering was not done properly, it makes logical sense that the requirements will change as the development of the system moves forward due to the lack of user involvement and missed requirements.

Another reason for changing requirements has to do with scope creep. Scope creep is defined as “the addition of requirements or functionality to the existing scope during execution of the project” (Thakurta, 2013). Thus, as the project is being delivered more functionality is added to the original scope of the project. This added functionality can pose major risks to the timeline and budget of the project, and if the risks are not mitigated and managed, they can cause the failure of a project.

2.6.4. Lack of executive support

Just how important is executive support in project success? The Project Management Institute (PMI) addressed this question in its 2017 *Pulse of the Profession* in-depth report: “Actively engaged executive sponsors continue to be the top driver of whether projects meet their original goals and business intent” (Project Management Institute [PMI], 2017). Executives have detailed knowledge of a project and how it ties up with the overall business strategy. They also have a position of authority to clear any impediments, make quick and effective decisions, and influence other executive leaders (PMI, 2017). The executive sponsor, in contrast to the project manager who

focuses predominantly on the day-to-day execution, is more strategic and focuses on creating conditions for success. He or she is crucial for leading major projects, especially those that cut across functions (Ashkenas, 2015).

The core group of people that should form the delivery team are (Harvard Business Review, 2016):

- Sponsor
- Project manager
- Team leader
- Team member
- Project steering committee

Not only does executive support play a vital role in the execution of the project, but it also plays a very important role in the accountability and ownership of the project. If there is no accountability and ownership of a project, it might lead to a runaway project that is over budget and that delivers functionality that is not needed. Thus, executive support plays a very important role in any project's success.

2.6.5. Technological incompetence

Technological incompetence occurs when an organisation does not have the necessary skills, or the availability of skilled resources, to deal with technological challenges that it is facing. This situation can transpire where companies and government adopt new technologies to solve problems or, in the case of companies, because competitors are using the technologies. This may happen when the company or government does not have the skilled staff necessary, nor access to the skills needed to implement the technology. The misconceptions accompanied by this approach are, firstly, the assumption that the competing company is successfully using the new technology and, secondly, the assumption that the proposed company has the resources required to implement the new technology (Anon, n.d.).

Should a company take on a project to deliver software without having the necessary competencies, it is setting the project up for failure. The project team will not be able to deliver due to the fact that they are not competent to do so.

2.6.6. Lack of resources

A company may not have all the resources needed to complete a project and thus have a lack of resources. These resources may be internal or external to the company. Resources are required to reach project objectives. The principal resource is skilled and competent staff. Other resources include capital, facilities, equipment, material, and information (Joubert, 2010).

Examples of resources needed:

- Human resources: Sponsor, project manager, analyst, developer, and tester.
- Capital: A project needs funds in order to, amongst other things, pay salaries to those involved.
- Facilities: Office space for the project team to meet and work from. This can include facilities to make food and drinks. Internet access also forms part of facilities.
- Equipment: This includes laptops and other hardware and software that are needed to perform the tasks.
- Information: Project teams need information relating to deadlines and project requirements.

If any of the required resources are missing, it places strain on the project team to complete a project on time and within budget. Resource constraints can lead to bottlenecks that can put a whole project at risk.

2.6.7. Unrealistic expectations

Should unrealistic expectations be set for the system, it can result in users of the system not receiving the results they expected. This can cause the project to be deemed a failure. In addition, because of unrealistic expectations, a system may need to be adjusted, resulting in late delivery.

Aggressive deadlines can lead to a team finding better and more efficient ways of doing work. However, unrealistic deadlines can result in missed delivery dates, going

over budget, poor quality, and low morale. The reason for this is that with unrealistic expectations generally come missed deadlines, for the obvious reason that these expectations were not attainable in the first place. Just because someone wants a project to take two weeks to deliver does not mean that it can be delivered within the desired time frame (Couvillion, 2013). “When unrealistic dates are imposed quality suffers as people cut corners and side step processes to meet deadlines” (Couvillion, 2013). Added to this is the rework that needs to be done due to the lack of quality and processes that were not followed properly.

Unrealistic expectations also lead to unhappy users of the system, due to the fact that they do not get what was promised to them. To mitigate the effect that unrealistic expectations can cause, the following actions can be followed (Couvillion, 2013):

- Understand the scope of work.
- Get estimates from the people who will be doing the work.
- Develop a basis of estimate.
- Re-estimate as soon as you realise an estimate assumption was wrong.

2.6.8. Unclear objectives

When the objectives of the project are not clearly defined at the start of the project, the project team has no direction or guidance. It is unwise to proceed unless the desired end results are clearly defined. Unclear objectives will never allow a project to achieve “done” status, as there will always be something outstanding (Herrick, 2011). In the case where the desired end product was never stipulated, the project team will not have anything to measure their progress by and will not know when they have reached their goal.

Research done by the PMI shows that 65% of organisations have a high alignment of a project to strategic goals (PMI, 2017). They recognise that great programme management delivers against strategy (PMI, 2017). Without clear objectives, it will be hard for a project team to track and measure if they delivered software on time, within budget, and with all required functionality. It results in wasted resources and man-hours.

2.6.9. Unrealistic time frames

Even with clear objectives, executive support, and all the resources needed to deliver the project within budget, the project is doomed to fail if the expected time frame for the project is unrealistic. One of the main reasons for unrealistic project time frames is the underestimation of effort required to complete the project. It might lead to poor design and defects (Ahmed, 2012).

Setting unrealistic targets may have far-reaching effects for a project team. There are short-term effects and long-term effects (Half, 2017):

- Short-term effects:
  - Missed delivery dates
  - Reduced work quality
  - Overrunning costs
  - Increase in absenteeism
- Long-term effects:
  - Low staff morale
  - Staff may aim lower
  - Loss of respect
  - High staff turnover

These effects can have a negative impact on any software project and can lead to project failures.

2.6.10. New technology

New technology can cause a project to fail if the skills and knowledge required to implement the technology are not available or understood. “Technology is always improving, which means there is a constant stream of new products being released over time” (Lee, 2013). With this continual improvement in technology come devices and software that are better and faster. However, those improvements come with compatibility issues (Lee, 2013).

When new technology is implemented that is not yet mature and all compatibility issues are not yet understood, it can lead to unforeseen issues. This can also happen when upgrading to the latest version of software that has not yet been tested. This was the case with Windows Vista, when many organisations upgraded only to find out that it was loaded with security issues and bugs (Asher-Shapiro, 2013). Implementing new technology poses a risk. There is an expectation of functionality without the security of knowing for sure if the system will work flawlessly (Lee, 2013).

2.7. Methods of delivering software projects

The methodology used to deliver a project also has an impact on its success rate. In cases where an Agile approach was used, increased success rates were reported in comparison with the more traditional Waterfall approach, where all analysis is done before development starts and testing only commences once development is completed. The Agile approach boasts a 39% success rate, compared to the Waterfall approach's meagre 11%. These results held true for projects of all sizes. The highest success rates for the Agile approach were seen on small projects, where 58% of projects were deemed successful and only 4% were deemed failures (Table 2.3).

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000

Table 2.3 Chaos resolution by Agile vs. Waterfall (Standish Group, 2015)

Based on these statistics, an analysis of the various methods and their success rates will follow.

2.7.1. Waterfall/Traditional

2.7.1.1. An overview of the Waterfall methodology

The Waterfall method is defined as the traditional approach to software development, where a project is broken up into distinct stages that must be completed in sequence (Shiotsu, 2018). Its name implies its workflow, where each stage represents a phase of software development, and one phase needs to be completed to move on to the next. Moving upstream is not allowed; once a phase is completed the full cycle needs to be completed before returning to the top. The phases of Waterfall are generally analysis, design, coding, testing, implementation, and maintenance (Figure 2.5).

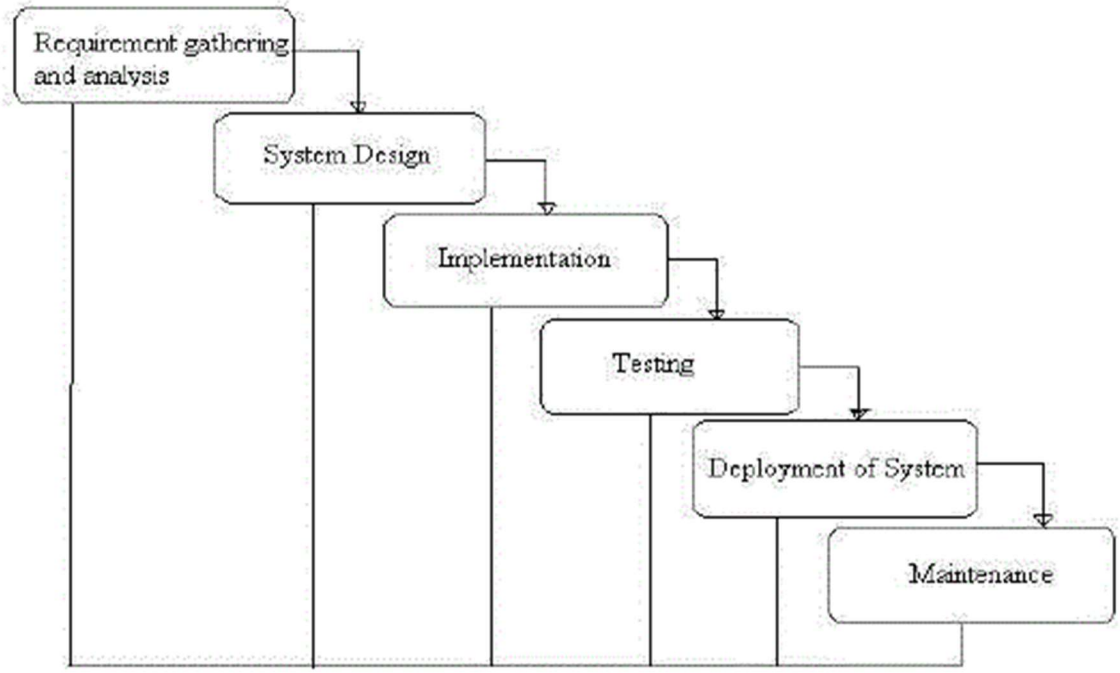

Figure 2.5 Waterfall model (Modi *et al.*, 2017)

2.7.1.2. Advantages of the Waterfall methodology

According to Kienitz (2017), the advantages of the Waterfall methodology are as follows:

- Clear deadlines and timescales: The phased development cycle enforces discipline. Each step has a clear starting point and conclusion, which makes progress easy to monitor.
- No financial surprises: Costs can be estimated with a fairly high degree of accuracy once the requirements have been defined.
- Testing is made easy: Test scenarios are already detailed in the functional specification of the requirements phase, which makes the testing process easier and more transparent.
- The outcome is crystal clear: Even before development starts, the design is ironed out in detail, which makes the needs and outcomes clear to everyone.

2.7.1.3. Deals with issues during the design phase: Potential development issues can be researched in the design stage – and alternative solutions planned – before any programming takes place. What you plan is what you get: Because everything is documented beforehand, there should be no surprises with the end product. Disadvantages of the Waterfall methodology

According to Shiotsu (2018), the disadvantages of the Waterfall methodology are as follows:

- Change can be costly. This is the major disadvantage to Waterfall; its rigidity hampers the ability to handle change. If one finds out late in the cycle that something does not work, or does not work as intended, it can be too late to change.
- Slow delivery time. Because the building of the system only happens later on in the cycle, stakeholders and users have to wait a long time before they see a working product.

- Gathering requirements too early is risky. Stakeholders and users often do not know what they really want until they have had a chance to see the system. Because all the requirements are gathered upfront, this can raise a real risk to the project.
- Tendency to neglect testing. Leaving all the testing to the end of the project can be a risk, because of the temptation to rush testing due to deadlines. This, in turn, can lead to a system that is full of defects.

2.7.1.4. When to use the Waterfall model

- This model is used only when the requirements are very well known, clear, and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements.
- Ample resources with required expertise are freely available.

2.7.2. Agile

2.7.2.1. An overview of the Agile methodology

The Agile methodology was put together as a solution to circumvent the pitfalls associated with Waterfall. Being a more flexible management framework, Agile allows teams to bypass traditional sequential paradigms and get more work done in a shorter time period (Chan, 2013).

Agile methodologies are a collection of values and principles that can be applied to a project. This methodology is more flexible than traditional methods, making it a better fit in a fast-changing environment. The Agile Manifesto (Highsmith, 2001) lists the following 12 principles to guide teams on how to execute with agility:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximising the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Agile takes an iterative approach to software development. Instead of planning upfront, Agile focuses on being lean and producing minimum viable products (MVPs) over set periods of time while improving with each iteration (Shiotsu, 2018). This model is fast becoming the project management method of choice, largely due to its focus on optimising development time and producing an operational application in the shortest time possible (Sweeney, 2017).

A common feature of the Agile methodology is sprints – miniature projects within the main project, usually divided up into two- to four-week increments. Each sprint is designed to produce a new working feature and is catered to developing the most essential parts of the application. This piece-by-piece approach allows developers to foresee and respond to all major obstacles and to set the project on a straighter course of development.

There are many different types of Agile development methods currently in use today. The most popular ones are Scrum, Kanban, and Extreme Programming.

2.7.2.2. Scrum

Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value (Schwaber & Sutherland, 2018). The Scrum team consists of a product owner, development team, and a Scrum master. Scrum teams are self-organising and cross-functional. The teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others that are not part of the team. The team model in Scrum is designed to optimise flexibility, creativity, and productivity.

There are a few prescribed events that are used in Scrum to minimise the need for meetings. These are:

- Sprint: A set time of less than a month during which a potentially releasable product is created.
- Sprint planning: The work that needs to be done is planned during sprint planning. This plan is done in collaboration with the entire Scrum team. It answers questions such as what can be delivered in a sprint and how the work to be delivered will be achieved.
- Daily scrum: This is a 15-minute event for the development team to coordinate activities and create a plan for the next 24 hours. It is held every day at the same time.
- Sprint review: This is held at the end of a sprint and attended by the Scrum team and the stakeholders to talk about what was delivered during the sprint.
- Sprint retrospective: This is an opportunity for the Scrum team to evaluate itself and find ways to improve team collaboration during the next sprint.

Scrum artefacts include the following (Schwaber & Sutherland, 2016):

- Product backlog: This is simply a list of everything that is known about and needed for the product. It is the only source of requirements for any changes

to be made. The product backlog evolves as the product and the features of the product evolve; it is constantly changing to cater for changing requirements.

- Sprint backlog: This is the set of product backlog items that was selected to be worked on in a sprint.
- Increment: This is the sum of all the product backlog items completed during a sprint.

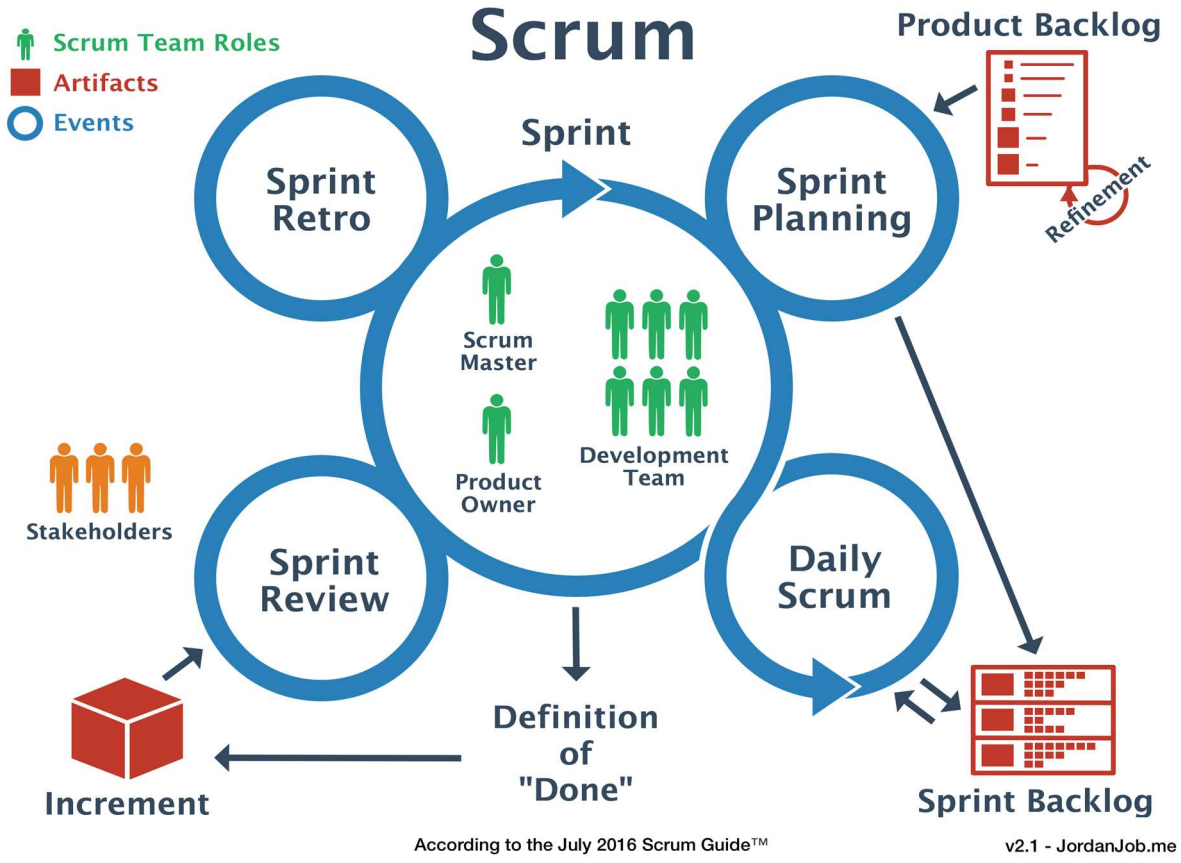

Figure 2.6 Scrum (Schwaber & Sutherland, 2016)

All these moving parts work together to form a structure that is capable of handling changing requirements on a continual basis while delivering value as early as possible (Figure 2.6).

2.7.2.3. Kanban

The Japanese word for “visual sign” or “card”, Kanban helps more traditional organisations improve their processes by visualising their workflow, limiting work in

progress (WIP), and enhancing the flow of backlogged items (Shiotsu, 2018). The system's highly visual nature allows teams to communicate more easily on what work needs to be done, and when. It also standardises cues and refines processes, which help to reduce waste and maximise value. Kanban visualises information by using sticky notes on a whiteboard to create a picture of the work (Figure 2.7). Seeing how this work flows within the team's process makes communication and context easy to relay.

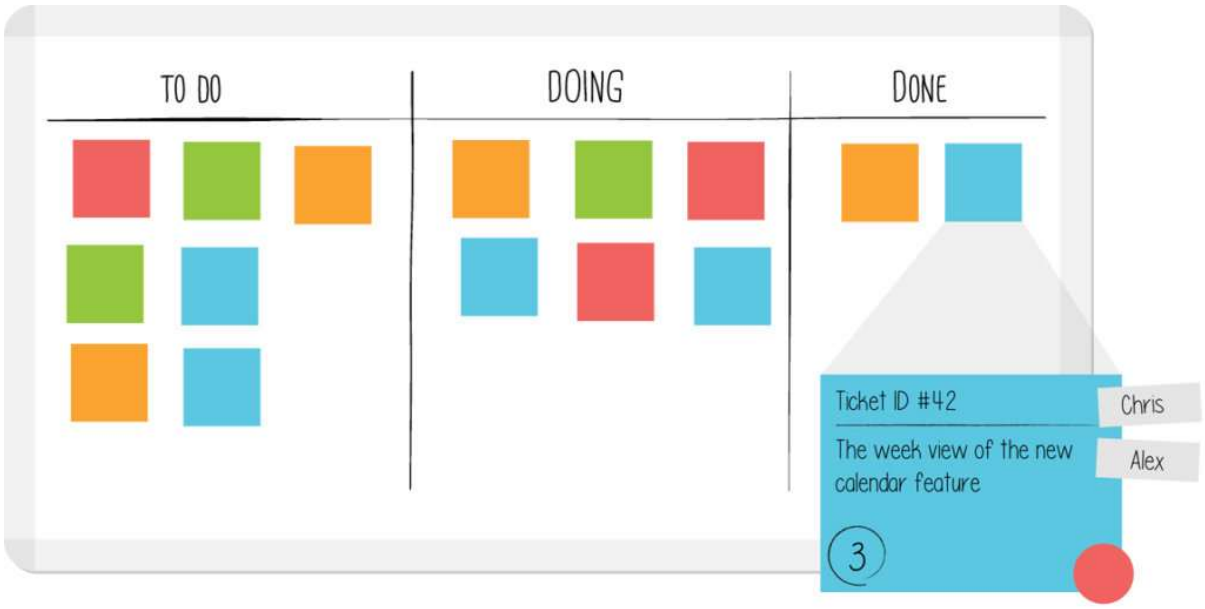

Figure 2.7 Kanban board (LeanKit, 2018)

Kanban works according to the following four core principles (LeanKit, 2018):

- Visualise work: By creating a visual representation of the work and how the work flows through the Kanban system, blocks and bottlenecks can be identified and resolved.
- Limit work in progress: By limiting the unfinished work in progress, the time it takes an item to travel through the system can be reduced.
- Focus on flow: Using WIP limits and policies can optimise the Kanban system to improve the flow of work. If the flow is not moving at the required rate, it can also indicate future problems.

- Continuous improvement: By tracking flow, quality, throughput, lead times, and more, the system and the team’s effectiveness can be continually improved.

By visualising the work in progress and streamlining the flow, the Kanban methodology can easily adapt to changes and deliver value on a continual basis.

2.7.2.4. Extreme Programming

Extreme Programming (XP) emphasises high software quality and responsiveness to changing customer requirements. Pair programming, extensive code reviews, and unit tests characterise this Agile methodology (Shiotsu, 2018). “Extreme Programming is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future, this process delivers the software you need as you need it. Extreme Programming empowers your developers to confidently respond to changing customer requirements, even late in the life cycle” (Wells, 2013).

XP improves a software project in five essential ways: communication, simplicity, feedback, respect, and courage:

- Extreme Programmers communicate with their customers and development team on a continuous basis.
- They keep their design simple and clean.
- They get feedback by testing their software starting on day one.
- They deliver the system to the customers as early as possible and implement changes as suggested.

Another characteristic of XP is programmers who work in pairs: two programmers at one screen doing extensive code reviews and testing all the code on a continuous basis. New product features are also avoided until they are actually needed. The whole idea is to put a minimal system into production as soon as possible and extending and growing it in whatever direction proves most valuable.

Figure 2.8 XP process (Wells, 2013)

By continually focusing on quality and making sure that the system is tested on a daily basis (Figure 2.8), change can be introduced by communicating to stakeholders and only focusing on what will add value.

2.7.2.5. Advantages of Agile methodology

The advantages of Agile methodology, according to Macaulay (2017), are:

- Faster delivery of working products.
- Rapid updates whenever needed.
- Closer collaboration between developers and the rest of the organisation breaks down working silos and encourages collective and individual buy-in through project ownership.

- Higher quality product due to focus on delivering and testing individual iterations.
- Faster detection and resolution of any issues.
- Easier ability to adapt as needs change and new ideas emerge.
- Greater focus on customer needs.
- Continuous improvement as the understanding of the project grows.
- Greater transparency of development processes.
- Provides the flexibility to introduce changes at any stage of the process.
- More efficient use of resources.

2.7.2.6. Disadvantages of Agile methodology

The disadvantages of Agile methodology, according to Macaulay (2017), are:

- Older organisations with long-established methods and team members who prefer a traditional approach may not be ready for dramatic changes to their working methods.
- Documentation can be neglected due to focus on working software.
- Close collaboration can be difficult to implement in teams spread across multiple locations or with stakeholders who are unprepared or willing to commit their time to the project.
- Deadlines can be difficult to implement on large projects due to the iterative nature of development.
- Agile methodologies can be hard to get the hang of compared to sequential approaches.
- The nature of iterative development makes it harder to accurately estimate costs and timeframes.
- Those who like to stick to firm plans may find that the final product is very different to what was initially expected.
- Developers have to dedicate a lot of time to the project due to the need for their ongoing and active involvement.

2.8. Which methodology is best suited for a project?

Waterfall should be used when the project has clear and exact requirements. There won't be any major changes in the scope throughout the project and the project is simple and predictable. Agile should be used when it is a new product and there is a need for speed, no real requirements upfront, and a need to reduce the cost of change and uncertainty.

2.9. Current situation

A study done in South Africa found that "the Agile method is the most dominant methodology adopted in the software development industry in South Africa and is also the preferred methodology to be used in the future. It is common practice for companies to adopt more than one software development methodology and the Waterfall method is the next methodology most widely used" (Ramnath, 2010). "Agile methods continue to be the predominant approach used for software development" (Reifer, 2017). Out of 150 organisations polled, 70% of the organisations reported using Agile methods, with Scrum being the most popular Agile method (Reifer, 2017). A full 71% of organisations are using Agile approaches for their projects. This has to do with an increased emphasis in many organisations on being more agile, customer-focused, and competitive (PMI, 2017).

With an increased uncertainty in business today it is clear to see that change is the only constant. That is why Agile methodologies are getting more attention in businesses compared to a decade ago. The ability of Agile methodologies to handle and embrace changes is making it the predominant method in South Africa and in the rest of the world.

2.10. Summary

The South African business environment is somewhat unique in the challenges that it faces with a high degree of cultural differences, high unemployment rates, and an economy that is growing at a very slow rate. The South African government is a huge contributor to the ICT sector, with billions of rands spent on projects annually. Not all of these projects are adding value to the community. The private sector is more

discreet with the number of projects that fail and the amount of money spent on projects.

This literature review of software projects and the reason for their failures indicate that studies have been done on the subject. The sources consulted varied in failure rates and reasons for failure. All the studies consulted conclude that software projects do indeed fail and it has emerged that there are some common reasons for these failures. According to the Standish Group (2015), 29% of projects are classified as successful, and 71% were either challenged or failed. The method of delivering the project also shows signs of different failure rates. Agile shows a 39% chance of success, compared to 11% chance of success for Waterfall. Although the Standish Group's list of 10 causes for project failure rates was used as the baseline for the literature review, there is a general agreement in the literature that these reasons are the most common.

In the following chapter, the information gathered in the literature study will be used to conduct a survey on South African businesses in order to determine whether South Africa is experiencing the same problems in software project delivery.

Chapter 3: Empirical study

3.1. Introduction

The preceding chapter reviewed literature relevant to this study. The aim of this chapter is to source responses from those in South African businesses who are involved in software projects. The layout of the questionnaire and the questions will be covered, the method of circulation and the target audience are explained, as is the response rate received. Lastly, the responses from the respondents will be presented in graphical form.

3.2. Data gathering

The Standish Group's 2014 and 2015 studies were used as the benchmarks for designing the research questions and for listing the reasons that software projects fail. These questions were posed in survey format to individuals working in South African businesses. Non-probability sampling was used, where individuals involved in software projects were identified through the researcher's social network and past software delivery experience. The main reasons for doing convenience sampling were as follows:

- The individuals' contact details were readily available.
- The individuals were known to be involved in software projects.
- Putting a survey link on social media was cost-effective.

As stated, the survey questions were designed based on Standish Group research that was done in the United States of America in 2014 and 2015. The aim of these questionnaires was to determine whether the findings hold true for the South African business environment.

The survey was created on the Survey Monkey website (www.surveymonkey.com), from where a link to the survey was generated which was then distributed on social media and sent to contacts. The cut-off date for the survey was 13 October 2017. The survey was also sent out via email wherever email addresses were available. A total of 23 individuals responded to the survey. Out of five emails sent, a total of three

recipients responded to the questionnaire. Twenty individuals responded on the social media platform. It could be concluded that people are more likely to responsive on social media platforms than they are on the more traditional email medium. However, 23 responses are not enough to draw valuable conclusions and thus the confidence level of the sample is not high.

3.3. The layout of the survey

The survey started with a cover letter giving the respondent the necessary information about the researcher and the reason for the survey. They were informed that the researcher is a student at the North West University who is in the process of completing an MBA dissertation. The nature of the survey was then introduced as follows:

“As you represent a company using software and operating in South Africa, I would appreciate your participation in assisting me in answering this short survey. When answering the questionnaire, please do so keeping in mind the most recent software project you were involved in, either as a user or as part of a project team that delivered it. These could include, but are not limited to, a website, office application, [enterprise resource planning] ERP solution, a custom-built solution, or out-of-the-box software that is used within your business. The questionnaire is totally anonymous, and any information will be kept confidential.”

The following 10 questions were asked in the survey:

Question 1: What type of industry is your company in?

This was a nominal-scale question asked in order to determine what industry the respondent works in, in order to try and cover as many industries as possible. The more industries were covered, the better the researcher would be able to draw conclusions on the South African business environment in general.

Question 2: What is your role with regards to software in your company?

This was a nominal-scale question asked in order to determine what the role of the respondent is. The main aim of this question was to try and cover all the roles that

were identified in the literature study, and to avoid getting answers from respondents only occupying one specific role. Responses from individuals who occupy a variety of roles would ensure a more reliable conclusion to the survey.

Question 3: Do you currently, or have you in the last two years, had any new software implemented, or were you responsible for implementing it?

This was a close-ended question to find out if the respondent is part of the target group.

Question 4: Was the software custom built or bought out of the box?

This was a close-ended question to find out if the respondent was part of the software development delivery cycle, as opposed to merely buying and implementing it. Again, this was to make sure the respondents were part of the intended sample group.

Question 5: Would you say that your last software project was a success (on time, within budget, and with all functionality)?

This was a close-ended question used to determine what the software project success/failure rates are in South Africa.

Question 6: What would you say are the major reasons for projects to be considered a failure?

This was an ordinal-scale question to determine what the respondents see as major reasons for failure.

Question 7: What would you say were the main reasons for project failure?

This was an ordinal-scale question based on the 10 main reasons why projects fail, as determined in the literature review. The aim of this question was to find out if the respondents are having the same problems and reasons for failure as the rest of the world.

Question 8: What would have added to the project being a success?

This was a nominal-scale question, based on the literature study, asked in order to find out whether the respondents are experiencing the same reasons for success as the rest of the world.

Question 9: What would you say would increase the chances of a project being a success?

This was an ordinal-scale question, based on the literature review, asked in order to find out whether respondents are experiencing the same reasons for success as the rest of the world.

Question 10: If you have any other comments or opinions regarding software projects within South Africa, please let me know.

This was an open-ended question, asked in order to elicit other responses and find out if the respondents have any other views that had not been discussed in the survey.

3.4. Data analysis

With the use of Survey Monkey comes the ability to gather data online and in real time. The responses were updated in real time and data analysis was done using statistical formulas in Microsoft Excel. Graphs and other statistical data were calculated and presented in report format. These results will be used in the next section to discuss the data that was gathered.

3.5. Results and discussion

3.5.1. Industry (Question 1)

The results show that most of the major industries are represented in the sample (Figure 3.1 and Table 3.1). The majority of the respondents are from the IT industry (30%), and second are finance and business services (17%). The only industry that is not included in the sample is manufacturing.

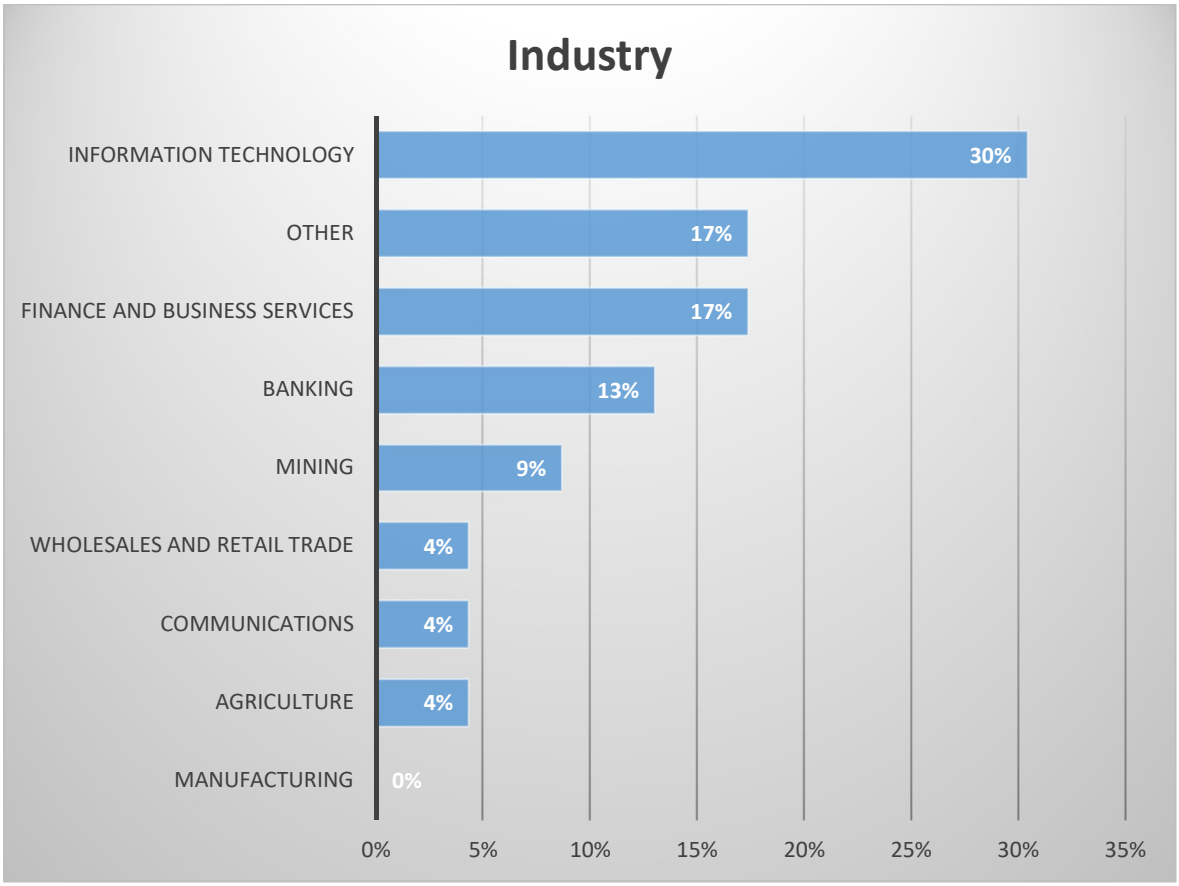

Figure 3.1 Industry

Manufacturing	0%
Agriculture	4%
Communications	4%
Wholesale and retail trade	4%
Mining	9%
Banking	13%
Finance and business services	17%
Other	17%
Information technology	30%
	100%

Table 3.1 Industry

3.5.2. Role (Question 2)

The results show that all the relevant roles as indicated in the literature review are represented in the sample (Figure 3.2 and Table 3.2). The majority of roles represented in the sample are developers (30%), followed by analysts (21%), then system users (18%), and finally product owners (6%), who play an important role.

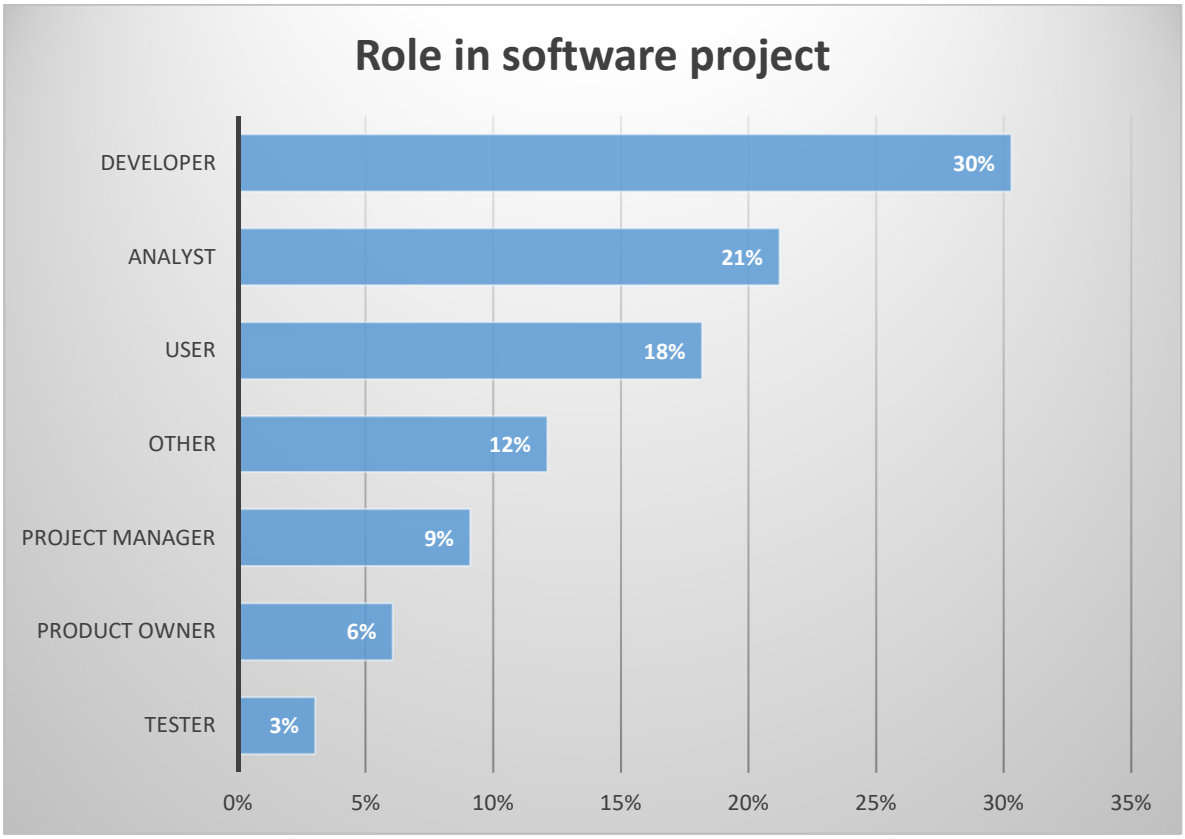

Figure 3.2 Role in software project

Tester	3%
Product owner	6%
Project manager	9%
Other	12%
User	18%
Analyst	21%
Developer	30%
	100%

Table 3.2 Role in software project

3.5.3. Involvement (Question 3)

A total of 96% respondents indicated that they have been involved in new software projects over the last two years (Figure 3.3 and Table 3.3). This is not really a surprise, as the survey was distributed to people in the IT sector.

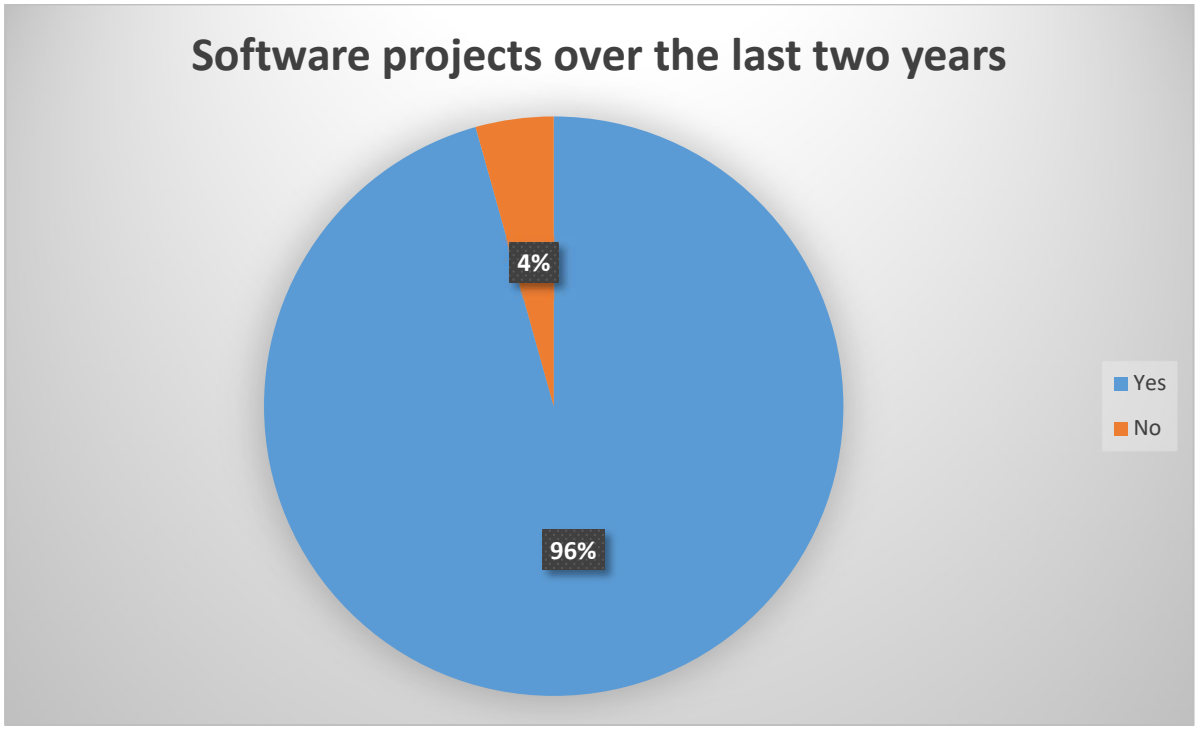

Figure 3.3 Software projects over last two years

Yes	96%
No	4%
	100%

Table 3.3 Software projects over last two years

3.5.4. Custom or out of box (Question 4)

Thirty per cent of the sample population indicated that they bought software out of the box (Figure 3.4 and Table 3.4).

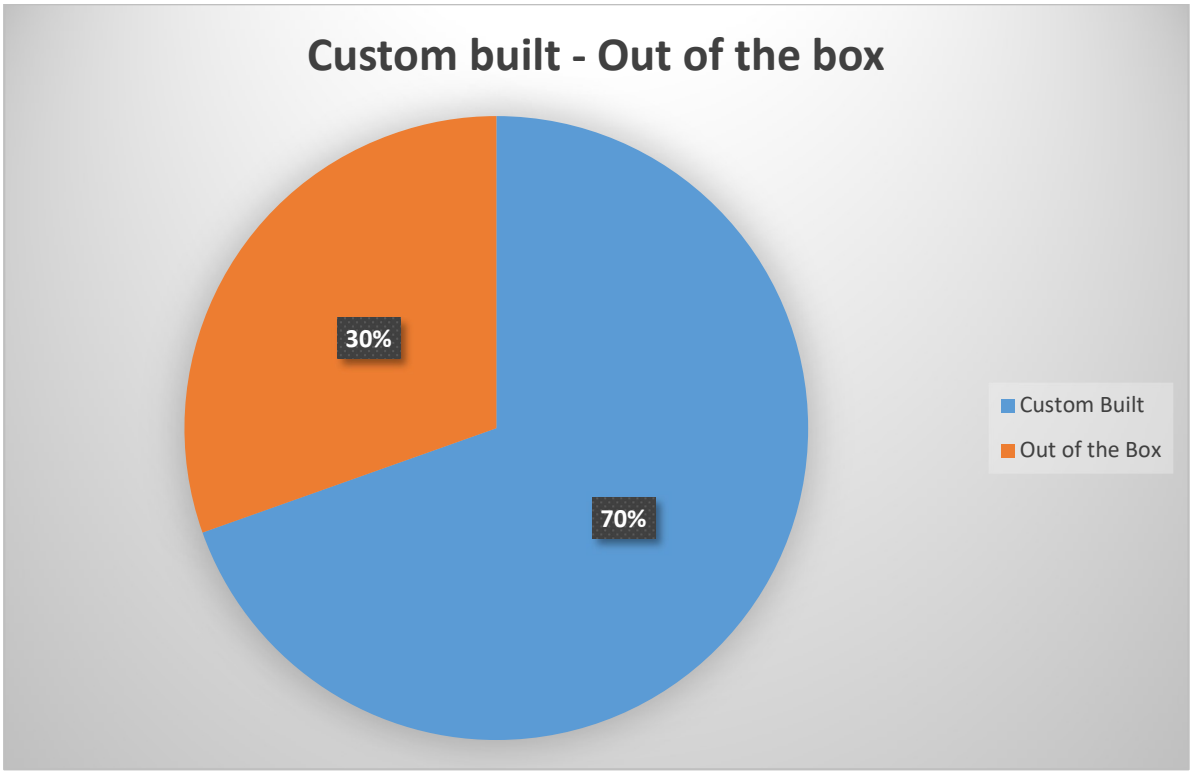

Figure 3.4 Custom built vs. out of the box

Custom built	70%
Out of the box	30%
	100%

Table 3.4 Custom built vs. out of the box

3.5.5. Success or failure? (Question 5)

The main finding of the survey is that 61% of the software projects that the respondents were involved in failed. This is 10% less than the 71% that the Standish Group reports for failed projects. According to the literature review, there are many different opinions on the percentage of failures, some putting it at 40 to 45%. One thing that all the literature consulted has in common is that the failure rate is always higher than the

success rate. This also seems to be the case in South Africa, as in this study only 39% of respondents indicated that their projects were successes.

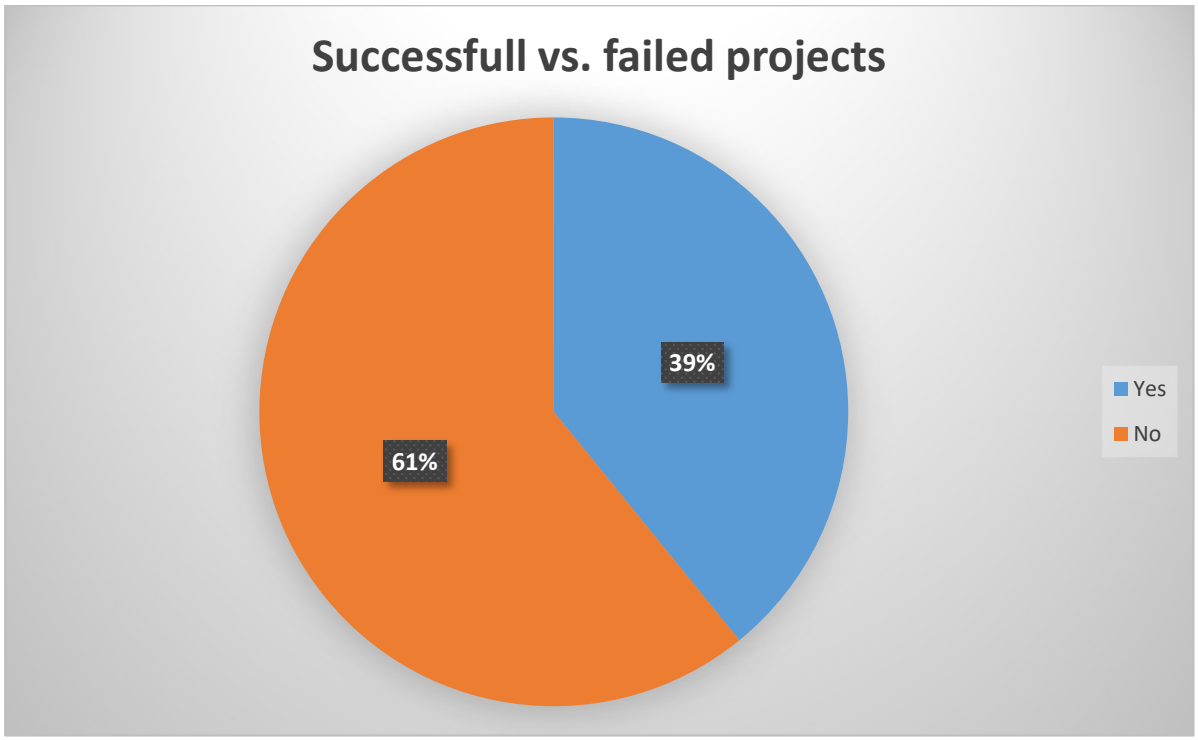

Figure 3.5 Success vs. failed projects

Yes	39%
No	61%
	100%

Table 3.5 Success vs. failed projects

3.5.6. Reason why project is considered a failure (Question 6)

Out of the sample group, the majority (22%) indicated that the main reason for a project to be deemed a failure is when not all functionality is delivered. The second biggest problem (17%) is late delivery. Fifteen per cent of the respondents indicated that projects failed because users were not using the system, and 13% pointed to projects being over budget (Figure 3.6 and Table 3.6). Of the top four reasons cited by survey respondents, three are listed in the definition of a failed project. Thus, the findings of this survey closely echo the findings in the literature, which define a failed project as being over budget, late, and when not all functionality is delivered.

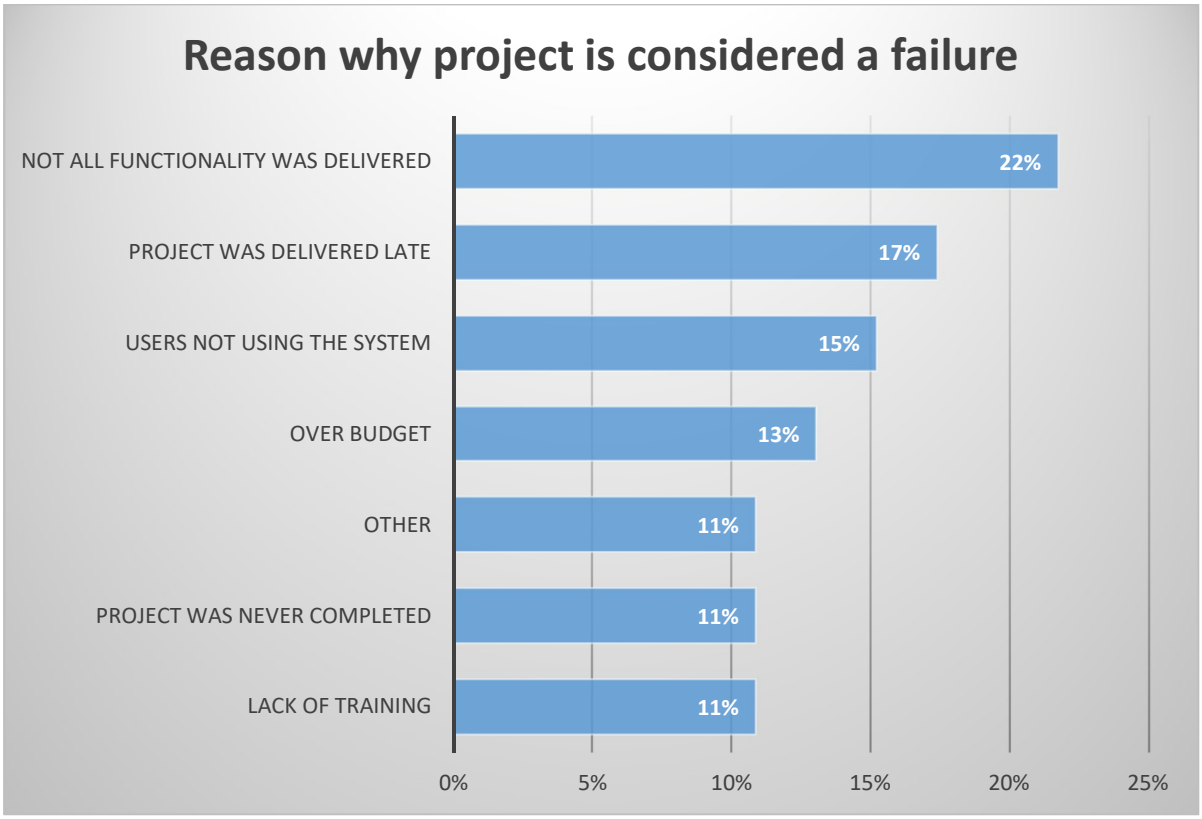

Figure 3.6 Reason why project is considered a failure

Lack of training	11%
Project was never completed	11%
Other	11%
Over budget	13%
Users not using the system	15%
Project was delivered late	17%
Not all functionality was delivered	22%
	100%

Table 3.6 Reason why project is considered a failure

3.5.7. Reasons projects fail (Question 7)

The findings from this question are echoed in the 10 reasons for project failure as per the literature review. The main reason for project failure as indicated by the study sample is a lack of user input from the start, with a weighted average of 4,14. Second is incomplete requirements and specifications, with a weighted average of 4,05. This is in line with the Standish Group's findings, which states lack of user input as the

number one reason for project failure, and incomplete requirements and specifications as reason number two.

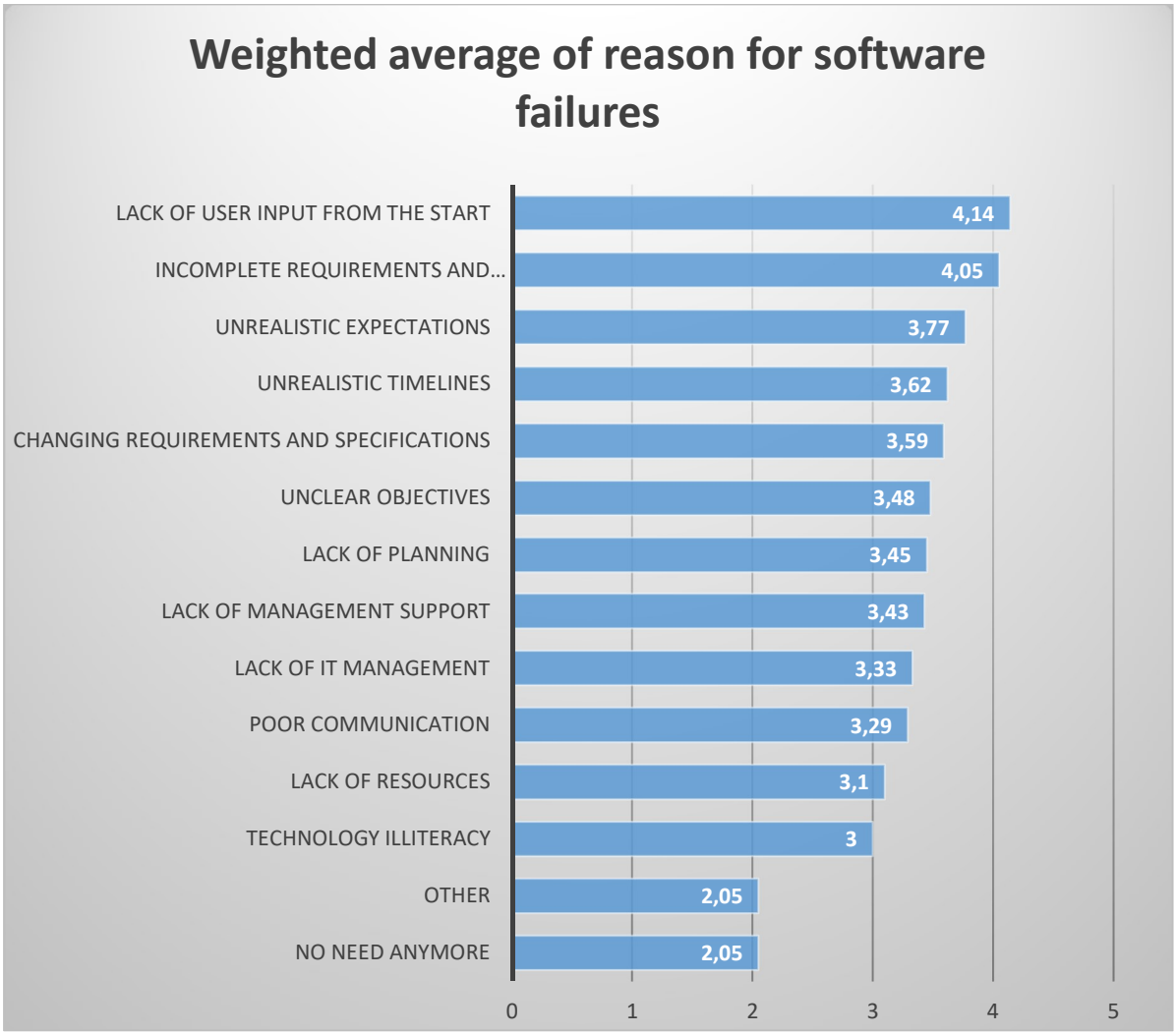

Figure 3.7 Weighted average of reason for software failure

	Not at all	Not really	Maybe	Probably	Definitely	Total	Weighted average
No need anymore	8	6	5	2	0	21	2,05
Other	9	4	5	1	1	20	2,05
Technology illiteracy	2	7	5	3	4	21	3
Lack of resources	3	4	5	6	3	21	3,1
Poor communication	2	3	6	7	3	21	3,29
Lack of IT management	2	4	5	5	5	21	3,33
Lack of management support	2	3	5	6	5	21	3,43
Lack of planning	2	4	4	6	6	22	3,45
Unclear objectives	5	1	3	3	9	21	3,48
Changing requirements and specifications	2	4	0	11	5	22	3,59
Unrealistic timelines	3	1	4	6	7	21	3,62
Unrealistic expectations	2	0	6	7	7	22	3,77
Incomplete requirements and specifications	1	1	3	7	9	21	4,05
Lack of user input from the start	2	1	2	4	13	22	4,14

Table 3.7 Weighted average of reason for software failure

3.5.8. What would have added to project success (Question 8)

The results show that 19% of respondents agree that effective communication would have added to a project's success (Figure 3.8 and Table 3.8). User involvement, and clear vision and objectives, scored 18%, in line with the Standish Group's findings.

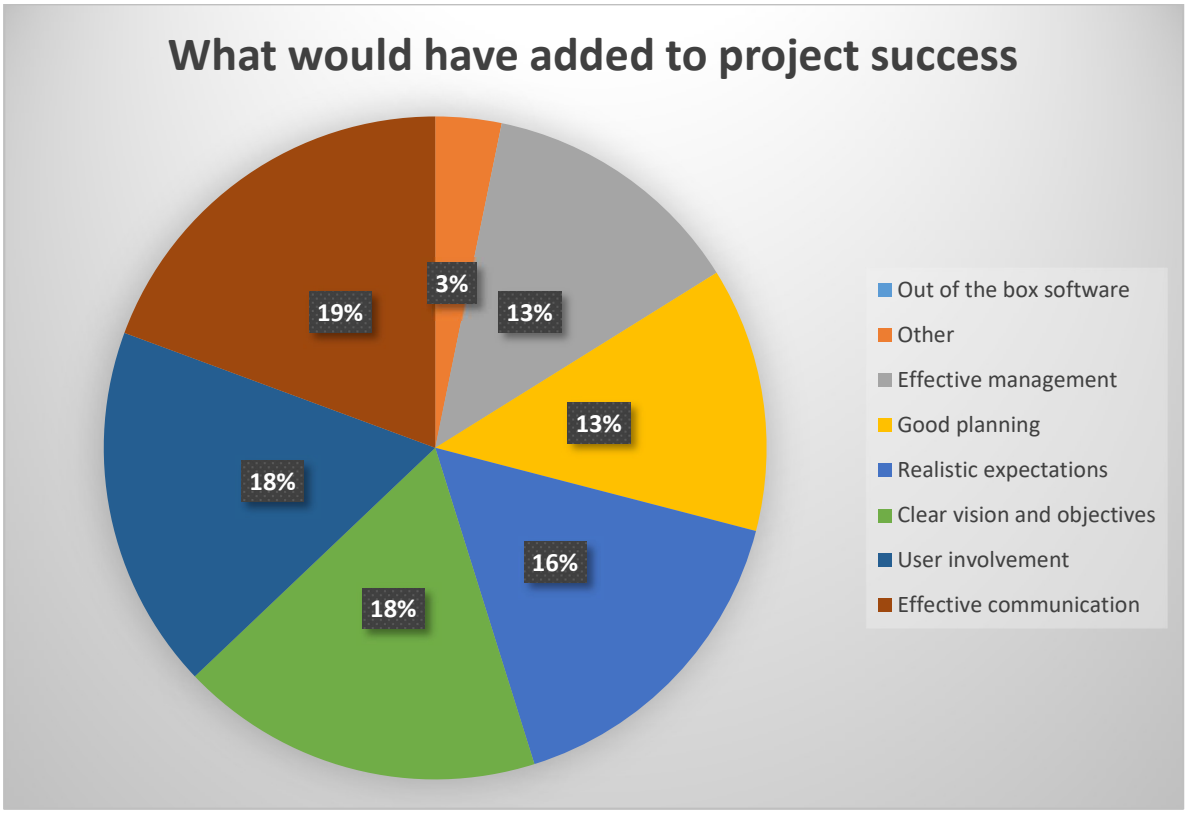

Figure 3.8 What would have added to project success

Out of the box software	0%
Other	3%
Effective management	13%
Good planning	13%
Realistic expectations	16%
Clear vision and objectives	18%
User involvement	18%
Effective communication	19%
	100%

Table 3.8 What would have added to project success

3.5.9. Increased chances of success (Question 9)

The results show that in terms of increasing a project's chances of success, good planning ranks first, with a weighted average of 4,72. Second is user input from the start, with a weighted average of 4,5. Again, user involvement appears in the top two responses, indicating just how important user involvement is to project success.

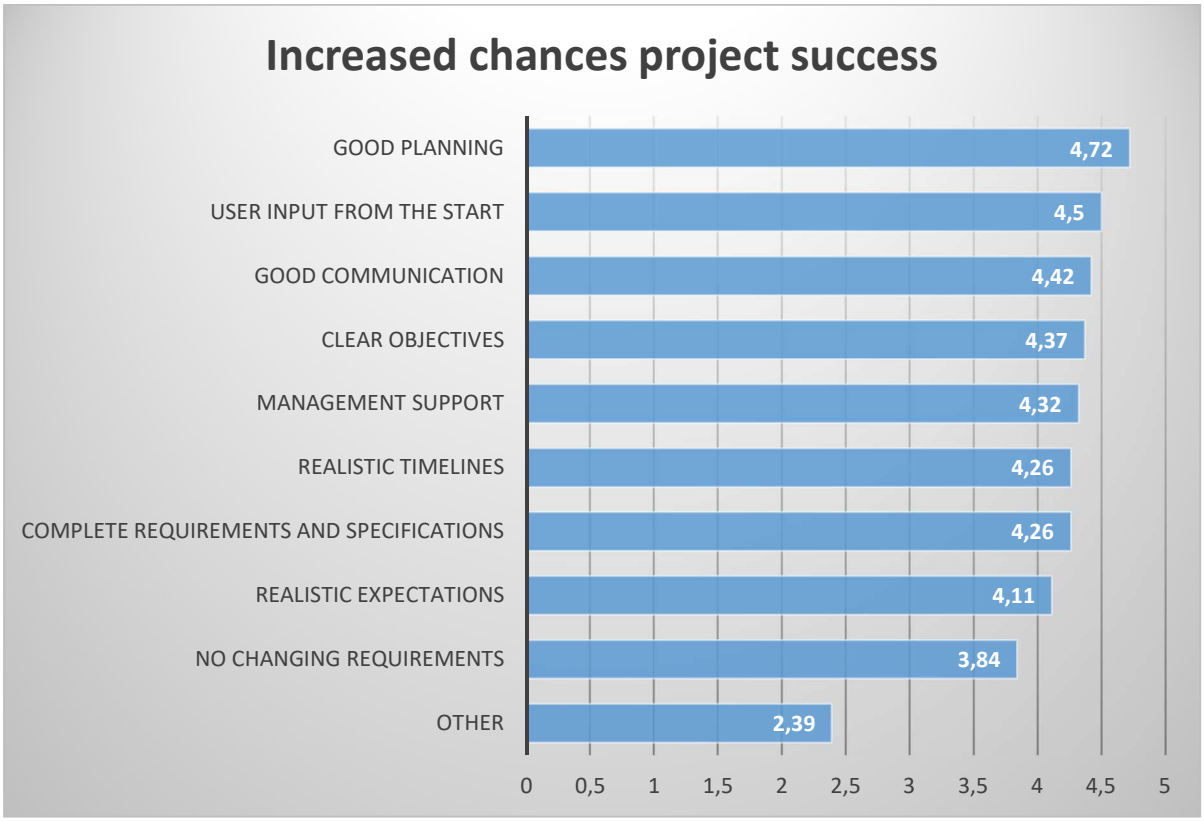

Figure 3.9 Increased chances project being a success

	Not at all	Not really	Maybe	Probably	Definitely	Total	Weighted average
Other	5	4	7	1	1	18	2,39
No changing requirements	1	1	3	9	5	19	3,84
Realistic expectations	0	0	4	9	6	19	4,11
Complete requirements and specifications	0	1	2	7	9	19	4,26
Realistic timelines	0	1	2	7	9	19	4,26
Management support	0	2	1	5	11	19	4,32
Clear objectives	0	0	2	8	9	19	4,37
Good communication	0	1	0	8	10	19	4,42
User input from the start	0	0	2	5	11	18	4,5
Good planning	0	0	1	3	14	18	4,72

Table 3.9 Increased chances project being a success

3.5.10. Open-ended question (Question 10)

Only eight respondents answered this open-ended question. Their responses indicate that:

- When sales are made, clients should understand that not all requirements are automatically catered for, and that certain requirements will need additional development.
- The involvement of the end users during the development process is very important, as they are the ones who will ultimately use the system.
- It is important to ask the operator before asking the manager.
- If the outcomes or deliverables of a project are properly planned and thus designed accordingly, realistic expectations and deadlines could be set. Sufficient buffer is typically incorporated into delivery dates in order to mitigate risks of unforeseen issues and testing. Properly planned projects and requirements, thoroughly documented and signed off by all parties, increase the potential for success. Changing any requirements (scope creep) affects the timeline and must be evaluated and justified by all parties. Project failures are primarily due to a lack of planning and documentation of deliverables and their corresponding deadlines.
- Software projects should be in line with clear business objectives. Those objectives contribute to the bottom line of the business.
- Getting input from users who use the system on a daily basis is important as they can give input as to what the system needs to do in the end.
- Distributed teams that do not work in same time zone can cause delays. To boost efficiency, all team members should work similar hours. All outsourced developers and analysts should be fluent in English. Project management and early communication with business-to-business partners are essential. Proper documentation is paramount, as there seem to be too many legacy systems with little or no documentation.
- A simplified dashboard and graphical user interface would be of benefit when working on complicated sheets. This will help to determine the share

of wallet and percentage contribution per stock keeping unit, as well as value contribution on national accounts.

3.6. International statistics from literature

In 2013 the Standish Group reported a 64% success rate for Agile methodologies, compared to a 49% success rate for Waterfall.

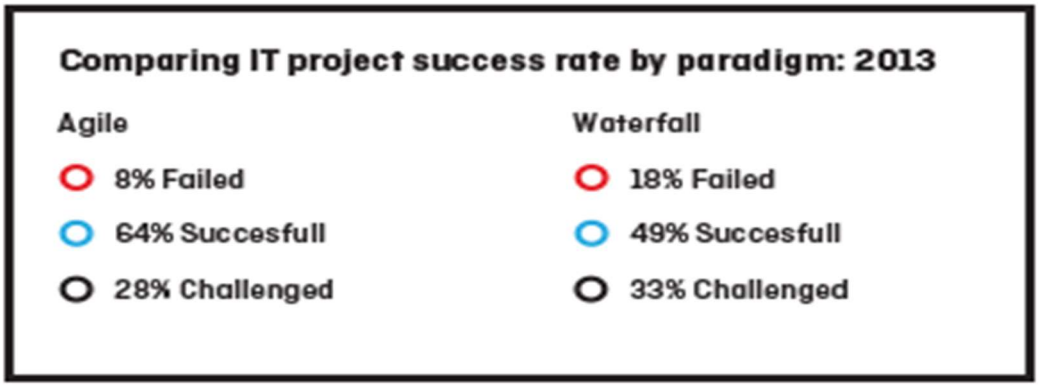

Figure 3.10 Comparing IT project success rate (Standish Group, 2015)

In 2015, the Standish Group showed a 42% success rate for Agile methodologies compared to the 14% success rate for Waterfall.

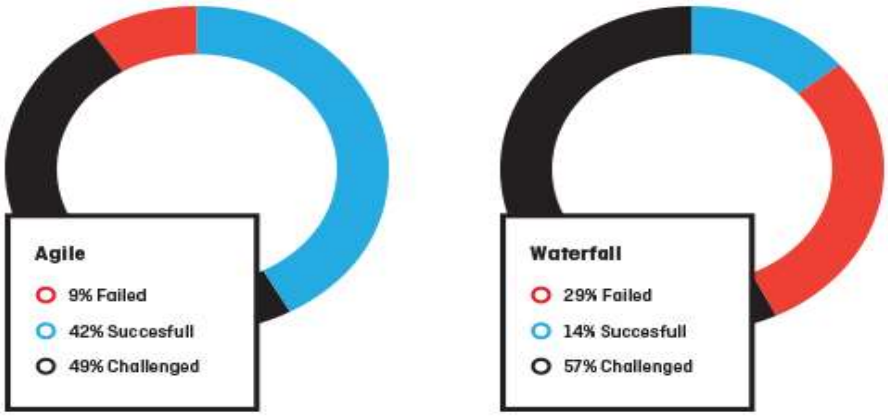

Figure 3.11 Agile vs. Waterfall success rates (Standish Group, 2015)

This shows a decline in the success rate of both the Agile and Waterfall methodologies. Key to note here is that the failed percentage of Agile has only increased by 1%, whereas the Waterfall methodology's failed percentage has increased by 11%. The fact that Agile is still under a 10% failure rate compared to Waterfall's 29% does indicate that, internationally, Agile methodologies have a better chance of successfully delivering a software project.

3.7. Summary

This survey of South African businesses has indicated that:

- 96% of respondents were involved in new software projects over the last two years.
- 61% of those projects failed.
- The main reason indicated for a software project to be considered a failure is that not all functionality was delivered (22%).
- The main reasons for software projects to fail include:
    - Lack of user involvement, with a weighted average of 4,14.
    - Incomplete requirements and specifications, with a weighted average of 4,05.
    - Unrealistic expectations, with a weighted average of 3,77.
- Factors that would have added to software projects' success:
    - Effective communication between the parties involved.
    - User involvement.
    - Clear vision and objectives.
    - Good planning and user input from the start.

The fact that a lack of user involvement is listed in the reasons for failure, and proper user involvement is listed as an indicator for success, points to the invaluable contribution of the user in project success. It stands to reason that if South African businesses want to increase the success rate of their software projects, they should focus on the user being involved from the start.

Chapter 4: Conclusions and recommendations

4.1. Introduction

The penultimate chapter was an analysis of the empirical study. In this chapter, conclusions and recommendations will be drawn on the basis of the literature review conducted in Chapter 2 and the survey analysis conducted in Chapter 3.

4.2. Conclusions

Businesses are faced with an ever-changing landscape. Technology is continually pushing the boundaries and changing whole industries. These changes result in uncertainty and challenges for businesses. These challenges bring with them risks that need to be managed. If these risks are not understood and identified they could lead to problems and substantial financial losses. Numerous studies have been conducted internationally to identify risks involved in software projects and to hone in on reasons for project failures. Studies show that software projects fail more often than they succeed. This is a cause for concern, as more and more companies are anticipated to embark on software projects in order to keep up with the growth in the internet and technology spheres. Hence it is imperative to gain a better understanding of the reasons why software projects fail so often. If these reasons are understood and acted upon, a better project success rate can be achieved.

The main purpose of this study was to explore the South African business environment in order to identify software project success/failure rates and potential causes for these successes and failures. This study found that, as far as software projects are concerned, South African businesses are experiencing the same challenges faced by international counterparts. Around the globe, software projects are reported to fail at a rate of around 70%. Comparatively, the surveyed South African businesses are experiencing a lower failure rate of 61%. Even though this is lower than the international market, the failure rate is still higher than the success rate.

The second purpose of this study was to determine common reasons for software project failure in South African businesses and to identify variables that could be added to the process to boost success rates. This study found that the reasons for project

failure as reported by South African business correspond with those of international companies. These reasons include lack of user involvement, with a weighted average of 4,14, incomplete requirements and specifications, with a weighted average of 4,05, and unrealistic expectations, with a weighted average of 3,77.

Two predominant methodologies are being used to deliver software projects, the new one being Agile and the more traditional, older one being Waterfall. These methodologies have their own individual strengths and weaknesses. These strengths can make one methodology more appealing than another for a certain type of project.

4.3. Recommendations

Based on the outcome of this study, several recommendations can be made.

- An analysis needs to be done to find out what methodology will work best for the project that is to be delivered, before development starts. If the requirements are well understood and are not expected to change during the course of the project, it will be best to use Waterfall as a delivery methodology. If the requirements may change and there is a lot of uncertainty, then an Agile methodology will work better.
- Users of the system should be involved from the onset of a project, especially when an Agile methodology is chosen.
- The user's input should be incorporated into the requirements and specifications of the project. These requirements and specifications need to be understood and agreed upon by all parties involved.
- Projects should be planned with realistic expectations in mind. Managing expectations within a realistic time frame will ensure a better outcome, as opposed to promising something that cannot be delivered in the time specified.
- Requirements may change as the project unfolds. There are different methodologies that can be used to cater for these changing requirements. An Agile methodology like Scrum might be better to use in order to deliver projects that are faced with changing needs.

- All projects should have a sponsor assigned to it. If there is no sponsor, there is little accountability.
- It would be wise to use an Agile development approach if requirements are likely to change. An Agile development approach has a higher success rate compared to the more traditional Waterfall approach.
- It is advisable to use technology that is tried and tested. Taking on new technology can put a project's success at risk.
- Drawing up realistic time frames during the planning process is of utmost importance. Setting unrealistic timeframes for projects have negative effects, which include:
    - Short-term effects:
        - Missed delivery dates
        - Reduced work quality
        - Overrunning costs
        - Increase in absenteeism
    - Long-term effects:
        - Low staff morale
        - Staff may produce lower-quality work
        - Loss of respect
        - High staff turnover

4.4. Achievement of the objectives of the study

Based on the literature review and empirical study presented in Chapter 2 and Chapter 3 respectively, the objectives of the study were achieved. The South African business environment has been studied and it was determined that software projects do fail. In addition, the main reasons for these failures were identified.

The second objective of the study was to determine which development methodologies could be used by South African businesses to help increase the success rate of software projects. According to the literature consulted, there are two main methodologies in use today that have various applications and success rates. It has been found that the Agile method of delivering projects does have a higher chance

of success. Companies in South Africa will thus have a better chance of success if they use an Agile method and involve users from the start of the project.

4.5. Recommendations for future research

This study has found that there are limited amounts of literature available that deals with the South African business environment, as the majority of the literature available is based on international research. This topic is rarely researched or studied in South Africa, and therefore merits further research locally.

Topics that need further research in the field of software projects within the South African business environment include:

- The true cost of failed projects.
- The definition of a failed project from the perspective of different stakeholders.
- Why South African businesses have a lower software project failure rate compared to international examples.
- The use of project readiness questionnaires and their potential to secure better software project success rates.

4.6. Summary

Chapter 1 outlined the research problem and contextualised the study. Chapter 2 consulted the literature to identify the main reasons why projects fail and what methodologies are currently in use, together with the advantages and disadvantages of each one. Chapter 3 presented the findings of the empirical study of businesses in South Africa, together with the key findings.

The aim of this research was to investigate the causes of project failures and to determine if these causes are present in South African businesses. Different methodologies were investigated to determine if there are methodologies with higher success rates than other. Through a literature review and an empirical study, the aims of the research paper have been achieved. The literature shows that there is consensus on the main reasons why projects fail internationally and in South Africa.

Agile methodology has proven to be more successful than the more traditional Waterfall methodology. The empirical study found that South African businesses are experiencing the same issues as international businesses.

Bibliography

Agarwal, N. & Rathod, U. 2006. Defining “success” for software projects: An exploratory revelation. *International Journal of Project Management,* 24(4):358-370.

Aljohani, M.D. & Qureshi, M.R.J. 2016. Management of changes in software requirements during development phases. *International Journal of Education and Management Engineering,* 6(6):12-26.

Ahmed, S. 2012. Casualty of unrealistic deadlines. *Project management in practice*. https://prince2msp.com/2012/03/20/casualty-of-unrealistic-deadlines/ Date of access: 16 Oct. 2017.

Ambler, S. 2010. Examining the agile cost of change curve. *Agile modelling*. http://www.agilemodeling.com/essays/costOfChange.htm Date of access: 18 Feb. 2017.

Anon. (n.d.). 12 deadly mistakes of software selection. *SoftResources*. http://www.softresources.com/resources/articles/12-deadly-mistakes-of-software-selection/ Date of access: 13 Oct. 2017.

Anon. 2011. Jo’burg billing system problems to face probe. *TechCentral*. https://techcentral.co.za/joburg-billing-system-to-face-probe/20713/ Date of access: 23 Sep. 2017.

Anon. 2016. Software development methodologies timeline. *Office Timeline*. https://www.officetimeline.com/blog/software-development-methodologies-timeline Date of access: 9 Apr. 2018.

Anon. 2018. 101 common causes. *International Project Leadership Academy*. http://calleam.com/WTPF/?page_id=2338 Date of access: 18 May 2018.

Asher-Shapiro, A. 2013. The drawbacks of adopting technology too early. *University Business*. https://www.universitybusiness.com/article/drawbacks-adopting-technology-too-early Date of access: 16 Oct. 2017.

Ashkenas, R. 2015. How to be an effective executive sponsor. *Harvard Business Review*. https://hbr.org/2015/05/how-to-be-an-effective-executive-sponsor Date of access: 17 May 2017.

Baron, J. & Spulber, D.F. 2017. The effect of technological change on firm survival and growth - evidence from technology standards. (Unpublished).

Beyer, H.R. & Holtzblatt, K. 1995. Apprenticing with the customer. *Communications of the ACM,* 38(5):45-52.

Bloch, M., Blumberg, S. & Laartz, J. 2012. Delivering large-scale IT projects on time, on budget, and on value. *McKinsey & Company*. http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value Date of access: 16 Feb. 2017.

BusinessVibes. 2015. The importance of information technology in business today. *Business2Community*. http://www.business2community.com/tech-gadgets/importance-information-technology-business-today-01393380#7TIXaZItvDB57p7G.97 Date of access: 4 Sep. 2017.

Chan, K. 2013. Scrum methodology vs. Agile methodology. *OneDesk*. https://www.onedesk.com/scrum-methodology-vs-agile-methodology/ Date of access: 3 May 2018.

Couvillion, D. 2013. Are you a victim of unrealistic expectations? *Impact Makers*. http://www.impactmakers.com/news/unrealistic-expectations/ Date of access: 13 Oct. 2017.

Department of Science and Technology ***see*** South Africa. Department of Science and Technology.

Edcon Holdings Limited. 2016. Edcon Annual Report 2016. https://www.edcon.co.za/pdf/annual_reports/annual_report_2016.pdf Date of access: 18 Aug 2017.

Ezer, J. 2010. Why do so many I.T projects fail? *The Huffington Post* http://www.huffingtonpost.com/jonathan-ezer/why-do-so-many-it-project_b_712060.html Date of access: 18 Feb 2017.

Farrow, J. 1997. Management of change: Technological developments and human resource issues in the information sector. *Journal of Managerial Psychology,* 12(5):319-324.

García-Duque, J., Pazos-Arias, J.J., López-Nores, M., Blanco-Fernández, Y., Fernández-Vilas, A., Díaz-Redondo, R.P., Ramos-Cabrer, M. & Gil-Solla, A. 2009. Methodologies to evolve formal specifications through refinement and retrenchment in an analysis–revision cycle. *Requirements Engineering,* 14(3):129-153.

Geneca. 2017. Why up to 75% of software projects will fail. *Geneca*. https://www.geneca.com/why-up-to-75-of-software-projects-will-fail/ Date of access: 18 Feb. 2017.

Half, R. 2017. The hidden risks of unrealistic expectations in the workplace. *Robert Half*. https://www.roberthalf.com.au/blog/hidden-risks-unrealistic-expectations-workplace Date of access: 16 Oct. 2017.

Harvard Business Review. 2016. Five critical roles in project management. *Harvard Business Review*. https://hbr.org/2016/11/five-critical-roles-in-project-management Date of access: 13 Oct. 2017.

Herrick, S. 2011. 3 dangers from unclear goals. *Cube Rules*. https://cuberules.com/2011/05/23/3-dangers-from-unclear-goals/ Date of access: 16 Oct. 2017.

Highsmith, J. 2001. Manifesto for Agile software development. *Agile Manifesto*. http://agilemanifesto.org/principles.html Date of access: 10 May 2018.

Institute of Directors in Southern Africa. 2009. King Report on governance for Southern Africa. https://cdn.ymaws.com/www.iodsa.co.za/resource/resmgr/king_iii/King_Report_on_Governance_fo.pdf Date of access: 11 Nov 2017.

Institute of Directors in Southern Africa. 2016. King Report on governance for Southern Africa. https://c.ymcdn.com/sites/www.iodsa.co.za/resource/resmgr/king_iv/King_IV_Report/IoDSA_King_IV_Report_-_WebVe.pdf Date of access: 11 Nov 2017.

IT Knowledge Portal. 2018. Software development methodologies. http://www.itinfo.am/eng/software-development-methodologies/ Date of access: 11 May 2018.

Pinto, J.K. & Slevin, D.P. 1988. Project success: Definitions and measurement techniques. *Project Management Journal,* 19(1):67-72.

Jørgensen, M., Mohagheghi, P. & Grimstad, S. 2017. Direct and indirect connections between type of contract and software project outcome. *International Journal of Project Management,* 35(8):1573-1586.

Joubert, P. 2010. Resource requirements for a project. http://www.peterjoubert.com/resource-requirements/ Date of access: 13 Oct. 2017.

Kelly, A. 2004. Why do requirements change? *Overload Journal,* 59. https://accu.org/index.php/journals/319 Date of access: 18 Jul 2017.

Kienitz, P. 2017. The pros and cons of Waterfall software development. *DCSL Software*. https://www.dcslsoftware.com/pros-cons-waterfall-software-development/ Date of access: 3 May 2018.

LeanKit. 2018. What is kanban? *Planview LeanKit*. https://leankit.com/learn/kanban/what-is-kanban/ Date of access: 11 May 2018.

Lee, J. 2013. 5 reasons why being an early adopter is a bad idea. *MakeUseOf*. http://www.makeuseof.com/tag/5-reasons-why-being-an-early-adopter-is-a-bad-idea/ Date of access: 16 Oct. 2017.

Mkentane, L. 2017. Pravin Gordhan under fire for IFMS programme. *Business Report*, 30 Aug. https://www.iol.co.za/business-report/economy/pravin-gordhan-under-fire-for-ifms-progamme-11008801 Date of access: 9 Sep 2017.

Macaulay, T. 2017. Agile project management CIO guide – The pros and cons of Agile methodology. *CIO*. https://www.cio.co.uk/it-strategy/cios-guide-agile-project-management-3669399/ Date of access: 11 May 2018.

National Treasury ***see*** South Africa. National Treasury.

Leke, A., Fine, D., Dobbs, R., Magwentshu, N., Lund, S., Wu, C. & Jacobson, P. 2015. South Africa's big five: Bold priorities for inclusive growth. *McKinsey & Company*. https://www.mckinsey.com/featured-insights/middle-east-and-africa/south-africas-bold-priorities-for-inclusive-growth Date of access: 18 Aug 2017.

Modi, H.S., Singh, N.K. & Chauhan, H.P. 2017. Comprehensive analysis of software development life cycle models. *International Research Journal of Engineering and Technology,* 4(6):117-122.

Nash, K.S. & Norton, S. 2016. Wal-Mart combines corporate IT and E-commerce technology groups. *The Wall Street Journal,* 15 Jan. https://blogs.wsj.com/cio/2016/01/15/wal-mart-combines-corporate-it-and-e-commerce-technology-groups/ Date of access: 18 July 2017.

Pillay, R. 2006. An investigation into the criteria for project success within Transnet. Durban: Durban University of Technology. (Dissertation – MBA).

Project Management Institute. 2017. Success rates rise: Transforming the high cost of low performance. *PMI's Pulse of the Profession.* https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf Date of access: 18 Apr 2018.

Ramnath, V. 2010. The level of adoption and effectiveness of software development methodologies in the software development industry in South Africa. Pretoria: University of Pretoria. (Dissertation – MBA).

Reifer, D.J. 2017. Quantitative analysis of Agile methods study (2017): Twelve major findings. *InfoQ*. https://www.infoq.com/articles/reifer-agile-study-2017 Date of access: 18 Apr 2018.

Sam Bad. 2014. Definition of project success. *StackExchange*. https://pm.stackexchange.com/questions/3122/definition-of-project-success Date of access: 13 Oct 2017.

Schwaber, K. & Sutherland, J. 2018. What is Scrum? *Scrum.org*. https://www.scrum.org/resources/what-is-scrum Date of access: 8 May 2018.

Shiotsu, Y. 2018. Agile vs. Waterfall: A side-by-side comparison. *UpWork*. https://www.upwork.com/hiring/development/agile-vs-waterfall/ Date of access: 3 May 2018.

Smith, J. 2012. Why information technology software projects fail in South Africa. Johannesburg: Technikon Witwatersrand. (Dissertation – MBA).

Sonntag, V. 2003. The role of manufacturing strategy in adapting to technological change. *Integrated Manufacturing Systems,* 14(4):312-323.

South Africa. Department of Science and Technology. 2007. Innovation towards a knowledge-based economy: Ten-year plan for South Africa (2008-2018). Pretoria.

South Africa. National Treasury. 2017. Budget Review 2017. Pretoria.

Standish Group. 2014. CHAOS Report 2014.

Standish Group. 2015. CHAOS Report 2015.

Statistics SA. 2016. The last 15 years: business income, spending and profit. http://www.statssa.gov.za/?p=9227 Date of access: 4 Sep. 2017.

Statistics SA. 2017. Three facts about ICT sector. http://www.statssa.gov.za/?p=9852 Date of access: 13 Nov. 2017.

Sun, Z. 2013. User involvement in system development process. Paper presented at the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.823.3265&rep=rep1&type=pdf Date of access: 11 May 2018.

Sweeney, M. 2017. Agile vs Waterfall: Which method is more successful? *Clearcode*. https://clearcode.cc/blog/agile-vs-waterfall-method/ Date of access: 3 May 2018.

Thakurta, R. 2013. Impact of scope creep on software project quality. *Vilakshan: The XIMB Journal of Management,* 10(1):37-46.

Verner, J., Sampson, J. & Cerpa, N. 2008. What factors lead to software project failure? Paper presented at the 2008 Second International Conference on Research Challenges in Information Science, Marrakech, Morocco, 3-6 June. https://ieeexplore.ieee.org/document/4632095/ Date of access: 11 Apr 2017.

Wells, D. 2013. Extreme programming: A gentle introduction. *Extreme Programming*. http://www.extremeprogramming.org/ Date of access: 11 May 2018.