DECISION SUPPORT FRAMEWORK FOR THE ADOPTION OF SOFTWARE DEVELOPMENT METHODOLOGIES

Dissertation submitted for the degree Magister Technologiae in Information Technology

In the Faculty of Applied and Computer Sciences

Lynette Simelane

209121807

Supervisor: Professor Tranos Zuva

Co Supervisor: Dr Etienne Alain Feukeu

DECLARATION

Student Number: 209121807

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

……………………….. ………………. Date

………………………………………… SIGNATURE

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof T Zuva, for his guidance throughout the study.

I would also like to thank all the participants for their invaluable input.

Most importantly I would like to thank my husband and children, for their support and encouragement throughout the study.

A special thanks goes to my mother for her support, encouragement; and all the sacrifices she made for us to receive quality education. I would also like to thank her for setting the bar and always reminding us of the importance of education.

Last but not least, praise be to God for granting me the opportunity to study and complete this degree.

TABLE OF CONTENTS

LIST OF ACRONYMS

ACRONYM	FULL NAME
CMMI	Capability Maturity Model Integration
EP	Extreme Programming
DSF	Decision support framework
FDD	Feature Driven Development
GSD	Global software development
IT	Information technology
JSE	Johannesburg Stock Exchange
OOAD	Object-Oriented Analysis and Design
PLS	Partial least squares
PM	Project managers
RAD	Rapid Application Development
SASD	Structured Analysis and Structured Design
SC	Software Crisis
SD	Software Development
SDLC	System development life cycle
SDC	Software development companies
SDM	Software development methodology
SDMs	Software development methodologies
SPI	Software process improvement
TAM	Technology Acceptance Model
US	United States
XP	Extreme Programming
DSS	Decision Support System
DS Framework	Decision support framework

ABSTRACT

There are many software development methodologies that are used to control the process of developing a software system. However, no exact system has been found which could help software engineers in selecting the best software development methodology (SDM). The increasing complexity of software development today has led to complex management of software systems.

This complexity increases the challenges faced by professionals in selecting the most appropriate SDM to adopt in a project. This is important because the wrong choice of methodology is costly for the organization as it may impact on deliveries, maintenance costs, budget projects and reliability.

In this study we propose a decision support framework to assist professionals in the selection of appropriate software development methodologies that would fit each organisation and project setting.

The case based reasoning (CBR) methodology was implemented in this study. This methodology focuses on problem solving that centres on the reutilization of past experiences. The CBR methodology was implemented using the SQL programming language.

We tested the precision of the decision support framework by comparing the recommended methodology to the actual software methodology that was adopted for the project. The DS framework recorded an 80% precision result. In addition, the findings contribute to reducing the software crisis faced by today's professionals. Therefore the framework can be adopted as a reliable tool for methodology selection in software development projects.

CHAPTER ONE

BACKGROUND INTRODUCTION

1.1 INTRODUCTION

Many software projects fail due to inappropriate selection of software development methodologies (SDMs). The increasing complexity of software developments today has led to complex management of these software systems (Despa, 2014) for many Information technology (IT) industries in the world (Dingsøyr *et al.* (2018)), particularly within South Africa (Silberberg & Africa, 2006). This complexity increases the challenges faced by managers in deciding the most appropriate software methodologies to adopt in a software project. This is important because the wrong choice of methodology is costly for the organization as it may impact on deliveries, maintenance costs, budget projects and reliability.

Whereas a right choice may minimize Software Crisis (SC) (Sharon *et al.*, 2010). SC was a period of software development misery claimed to have tormented computer science since early 1970s (de Vasconcelos *et al.*, 2017). According to Haigh (2010), SC is attributed to a lack of availability of a functional decision support framework (DSF), amongst other factors. Likewise, SEI (2006) affirms that frameworks can assist organises to improve in the way it operates. Making DSF vital in software development, hence, the availability of DSF for the adoption of software development methodologies will provide project managers with the necessary knowledge to gain insights and take decisions driven by the framework. Additionally, DSF will be a means of solving the challenges of selecting the right SDMs and tackling the existing software crisis.

SC was caused by the need to increase quality, tools and methods for software engineering. Software methodologies assisted in organising the software development process from documentation of requirements through to maintenance and support of the end product. A software development process involves a group of activities, methods, tasks, actions and practices that are used within the process of creating a high-quality software product (Georgiadou, 2003). In this study, the focus is on SDM and it refers to the framework used to structure, plan and control the process of developing an information system (Centers for Medicare & Medicaid Services, 2008). There are many different methodologies used to develop software products e.g. waterfall, incremental process, prototype, spiral, iterative, and RAD, to mention a few.

Additionally, software development methodologies (SDMs) is a way of managing a software development project which typically address issues like selecting features for inclusion in the current version, when software will be released, who works on what, and what testing is done. While SDMs have been scantly covered in some of the IT degree curriculums, individuals working for professional software development organizations find that it is a big part of their daily work environment. When selecting a SDM, the user is confronted with an assortment of possibilities and the difficult task of identifying which method is most suited to the salient characteristics of the decision situation, and the environment in which it exists. Most of these SDMs are based on three generic methods, namely: the sequential methods, the evolutionary (iterative) methods and the component-based software methods (Sommerville, 2016).

Most software processes available currently are based on these methodologies, there is no one size fits all methodology, as none is considered as ideal or always the best (Kuhrmann *et al.*, 2018)& Mahapatra, 2015). Since there are many methodologies available, one of the challenges faced by project managers is to decide on the most suitable methodology to adopt in a software project. For many organizations, the analysis of all processes and the characteristics of each new project is time consuming. Hence, most organizations consider choosing software development methodologies known to them and at the same time familiar to all employees as a suitable decision.

Unfortunately, according to Sharon *et al.* (2010) this manner of selecting SDMs is only suitable for medium to large systems. From reviewing over a hundred software development projects in organizations, Georgiadou (2003), found that almost 80% of projects were unsuccessful because these organizations apply methodologies to projects without putting into consideration the characteristics of these projects and other methodologies available. According to Georgiadou, the success of a software project should be heavily dependent on the characteristics of the project and method in use. Likewise, Meulendijk and Oud (2007), claimed that the most suitable methodology depend on the characteristics of the particular project. Similarly, Vliet (2008) argued for the need to apply a certain software methodology grounded on the characteristics of the project.

One possible means to cope with the above is to tailor a method according to the organization and its environment. Unfortunately, this strategy will demand a need for project managers to continually monitor, analyse and adjust the method, tailoring strategies and the project environment during the project (Xu & Ramesh, 2008). Using different methodologies for different projects may be another solution. By providing project managers a decision support framework for selecting the right methodologies, valuable time is saved. Besides, projects that are adopted with the use of a suitable development methodology, may increase the quality of the final product.

Therefore, the benefits and significance attached to decision support framework cannot be overemphasized. Developing a decision support framework is a means of tackling the existing challenges of selecting the right SDM. Selecting an appropriate SDM can make a big difference in achieving a successful end result when measured in terms of cost, meeting deadlines, client happiness, robustness of software, or minimizing expenditures on failed projects. It is against this background that this study attempts to evaluate the methodologies applied and practiced on various software development projects with the overall objective of developing a decision support framework for the adoption of software methodologies. This will ensure successful completion of business critical software development projects and realization of business objectives for which the projects were undertaken (Khan & Beg, 2013).

1.2 RATIONALE AND MOTIVATION OF THE STUDY

Previously, software development consisted of a programmer writing code to solve a problem or automate a procedure. Nowadays, systems are so large and complex that teams of architects, analysts, testers, programmers and end users must work together to create a solution that satisfies business needs and requirements (Mahanti et al., 2012).

Organizations constantly adapt their information systems to reflect changes in the type of information needed because of changes in technology, business processes, structure, or the external environment. A process called the systems development life cycle (SDLC) has been developed to ensure that these changes are orderly and productive. To manage this, a number of system development life cycle (SDLC) methodologies have been developed, namely: waterfall, spiral, RAD, prototyping and JAD. Beside the existence of the SDLC, and based on the vast varieties of available methods, it is still very difficult to choose appropriate methodologies which will play an important role in ensuring that the project is delivered within schedule, within cost and meet the user requirements.

1.3 PROBLEM STATEMENT

Some concerns in the world have indicated a need for a methodology that is cost effective and results in higher quality and success rate (Overhage et al. 2011). For example, a report from the Standish Group CHAOS (2015) revealed that most projects run globally in 2015 were either challenged (52%) or failed (19%) (Standish Group 2015 Mahapatra, 2015).

The software development industry emerged over six decades ago (Cusick et al., 2008). Since the late 1960s, lots of methodologies have been developed and introduced to try address the challenges that emerge during the software development life cycle (Jiang & Eberlein, 2008). Presently, there are various models, standards, procedures, methodologies and process improvement guidelines that have been introduced to assist an organisation in the way it operates and conducts business (SEI, 2006).

A series of common software development methodologies is summarised by Griffin and Brandyberry (2010) as depicted in Figure 1 below.

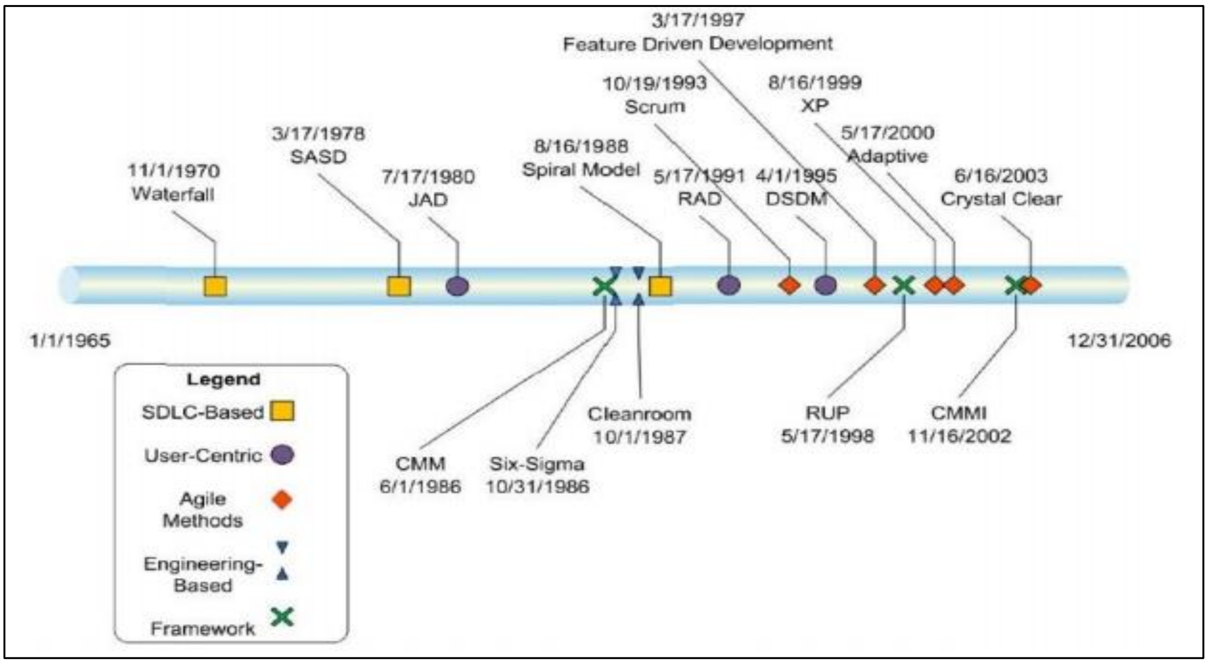


Figure 1: Software Development Methodology Time (Griffin & Brandyberry, 2010).

Failures in software development projects have attracted much attention in academia, industry and the government because of the on-going failures of software development projects in relation to meeting time, budget and functional requirements (Marques, Costa, Silva, & Gonçalves, 2017). As such, failures in software development projects have dominated the topic of several studies (Abrahamsson, Salo, Ronkainen & Warsta, 2017; Batarseh & Gonzalez, 2018; Lei, Ganjeizadeh, Jayachandran & Ozcan, 2017; Varajão, J. et al., 2014). Unfortunately, these studies are limited, in that they focus more on project management and significantly ignore the kind of software development practices suitable for each organizational and project setting. It was also pointed out by scholars (Dingsøyr *et al.* 2018) that implementing software that is within budget, schedule, satisfies customer requirements and is of good quality, seems to be a very big challenge for professionals in the field. Therefore, in closing this gap, the present study develops a decision support framework to assist in the selection of appropriate software development methodologies that would fit each organization and project setting.

1.4 Research Questions and Hypotheses

This study is designed to provide a solution to the research question below:

"How to determine the best methodology to adopt in a software development project?"

Below are the formulated sub questions for this research. These sub questions are an answerable inquiry into the specific concern. They relate to the different components of this research as stated below:

- How to propose a suitable tool for selecting the most appropriate methodology to adopt in a software project?
- To what extent do the project characteristics inform the adoption of the best fit software development methodology?
- How to measure the effectiveness of the proposed decision support framework for the adoption of software development methodologies?

1.5 Research Objectives

This study is aimed at developing a framework to assist organizations in making a sound decision regarding which software development methodology to apply during their software development lifecycle.

1.6 The Objectives of the Study

- To study what has been done in literature in relation to adoption of appropriate software development methodologies and the factors informing the decision.

- To propose a decision support framework to assist in the selection of a suitable software development methodology to adopt for a particular project.
- To evaluate the proposed framework by demonstrating its effectiveness in a real world environment and also by comparing the results to that of existing evidence by other researchers.

1.7 OUTLINE OF THE STUDY

This study is structured in five chapters following the V-Model (Sheffield, 2005), This model describes the various chapters of the study and how they relate to each other. The intention or purpose of the research appears on the left, and the results and outcomes appear on the right. This model ensures alignment between purpose and outcomes of a study.

This first chapter describes the problem and motivation of the topic of interest. It also presents the research objectives, questions, and the chapter outline.

The second chapter looks at previous literature in the field; this is important for studying the gaps and developing a research model for the study.

Chapter three informs the reader about the research philosophy and methods that were used to gather data and evaluate the research model. Furthermore, a hypothesis is developed. This chapter provides a basis for the data analysis as presented in chapter 4.

Chapter four reports on the analysis of data gathered as per method described in chapter 3. The results of statistical analyses of the data as well as illustrations are presented, interpreted and discussed in this chapter.

The final chapter concludes this study with a summary of the key findings and qualified responses to the initial research questions. It also discusses the limitations of the study and recommends further research if required.

1.8 SUMMARY OF THE CHAPTER

The chapter discussed the background introduction of software development methodologies, and considered the problem statement, which was the departure point for the study. This was substantiated with the rationale and a motivation why the study needed to be undertaken.

Thereafter, research questions, hypothesis and study objectives were clearly stated, followed by the chapter outline.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter consists of an analysis of the characteristics of software development and the theoretical analysis of software development methodologies. The chapter presents analysis of current software development methodologies. These methodologies are presented by emphasizing, benefits, strengths and weaknesses from the user's perspective. This is done in order to understand reasons for adopting and selecting these methodologies. Thereafter, relevant literature review on project characteristics and the adoption of software development methodologies was discussed. Likewise, the chapter review literature on the extent to which organisational factors influence the way software development is carried out in software enterprises. These reviews are important to attempt to identify gaps in previous studies and the need for developing a decision software framework in order to ensure successful completion of business (Khan & Beg, 2013). The chapter ends with a summary, this provides an overview of the literature reviewed, by tabulating various research topics, methodologies, findings and gaps identified in the various study reviewed.

2.2 OVERVIEW OF SOFTWARE DEVELOPMENT METHODOLOGIES

The main purpose of this section is to provide an overview of different software methodologies so as to explain the evolution of software methodologies. This is relevant as it will set the rationale of this study.

In the last three decades, various software development methodologies have been developed, but only a few remain relevant today (Abrahamsson *et al.*, 2017b; Despa, 2014; Tavares *et al.*, 2017). This is because the software industry is forever developing and introducing new and improved methodologies, particularly because organizations are constantly adapting their information systems to reflect changes in the type of information needed as a result of changes in technology, business processes, structure, or the external environment (Zykov, 2018). The change is informed by the increasing challenge of identifying and selecting the best-fit, appropriate software development methodology. Besides, the need to improve quality, reduce cost and develop faster software is overwhelming for project managers who are continually faced with the task of choosing from numerous methodologies and variations of methodologies to match the needs of the software project or the nature of the organisation.

Nevertheless, a process called the systems development life cycle (SDLC) was developed to ensure that these changes in technology and business processes are orderly and productive. To manage this, a number of SDLC methodologies have been developed, namely: waterfall, spiral, RAD, prototyping and JAD. The oldest of these, and the best known, is the waterfall: a sequence of stages in which the output of each stage becomes the input for the next. These methodologies are examples of the traditional methodologies, the next section detail an overview of the traditional methodologies.

2.2.1 The Traditional Methodologies

Traditional methodologies can be described as predictive, process-focussed, document and plan driven (Boehm, 1988; Despa, 2014), and follows sequential steps (Paul *et al.*, 2008). The methodologies include thorough planning, formalised processes, proper documentation, and sustainable design processes (Despa, 2014). Additionally, traditional methodologies requires software initiatives to begin with the gathering of requirements, and documentation thereof, followed by an architectural design, development and testing (Awad, 2005). It has been suggested (Gill *et al.*, 2018) that traditional plan-driven software development practices (e.g. waterfall, spiral) may be more applicable in large projects.

Paul *et al.* (2008) emphasizes that the Waterfall model is widely used in smaller software development companies because it is simple and easy to understand. This implies that organization size and effort of implementation are factors that have an influence in methodology selection.

Equally, traditional methodologies emphasize that complex systems can be built in a single pass without necessarily revisiting or changing requirements (Victor Szalvay 2008) . Thus, solid understating of the stakeholder's needs are vital according to (Paul et al. (2008), which is not the case in today's complex and high-technology environment (SEI, 2006). Thus, indicating another factor influencing methodology selection is the complexity of requirements, identified as a limitation of the Waterfall model.

The spiral model is refined with basic principles internalized within the Rational Unified Process (RUP), extreme programming, and generally the agile software development framework (Paul *et al.*, 2008). The spiral model is not without shortcoming, as it is noted that spiral methodology has challenges in identifying the right moment to move onto the next phase.

More so, it is only those businesses that are uncertain about their requirements or expect major edits in their mid to high-risk project that majorly benefit from the scalability of spiral methodology. These weaknesses are reasons for the evolvement of more iterative development methodologies Paul *et al.* (2008). Nevertheless, the Waterfall model is still widely in use particularly within the context of South Africa (Paul *et al.*, 2008 & Vinekar *et al.* 2006). Consequently to this, the Waterfall model is analysed briefly below followed by a graphic representation of other traditional methodologies. This is done to better individualize and visualize their structure.

2.2.1.1 Waterfall Model

The waterfall method involves several phases and each phase includes multiple steps as illustrated in Figure 2 below. The model was named waterfall due to the sequential manner that one process follow after another. The model is so designed that the next phase cannot begin unless the previous phase has been completed and approved. The advantage of this process is that it is orderly and team members are aware of the work required and when they may proceed to the next steps (Despa, 2014).

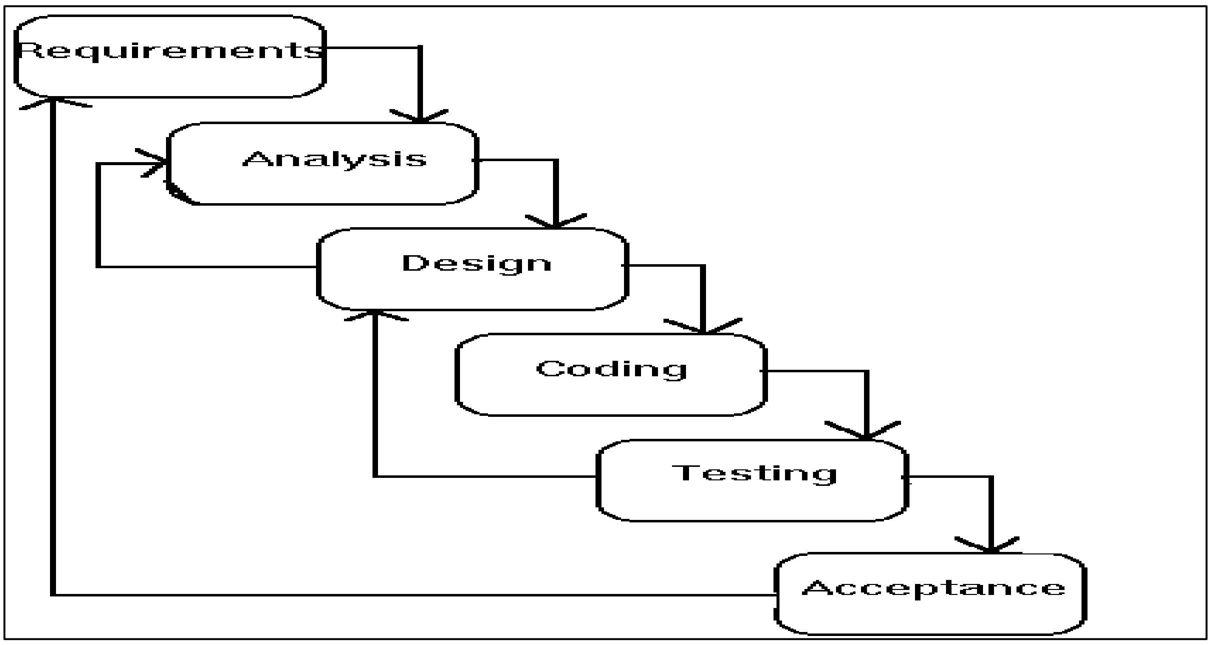


Figure 2: Waterfall Methodology (Dr Winston W Royce 2016)

Though software projects do not need to always follow a strict linear process, this may be because clients' requirements often change or may be misinterpreted. Moreover, rules can be adjusted, likewise, through project analyses certain issues become clearer with time. According to Vliet (2008) the Waterfall model is unrealistic, arguing further that, in many projects, real sequential steps are often not obeyed. In the same manner, over 4 decades ago, Dr Winston W Royce (2016) asserted that the Waterfall model is risky and prone to failures. The most issues are experienced due to the use of this type of iterations. For instance, if the project is in testing and errors/bugs are identified, a major redesign is required to fix the issues. This inflexibility causes the developers to stall the parts where errors are found. For instance, if problems have occurred in system requirements or analyses they are left for later resolution. This results in problems being ignored and obviously causes poor solutions that do not conform to the user's requirements/wishes (Sommerville, 2016).

2.2.1.2 Spiral method

The spiral methodology permits team members to adopt multiple SDLC models based on the risk patterns of the given project. A blend of the iterative and waterfall approaches, the issue associated with the spiral model is identifying the right moment to move onto the next phase as depicted in Figure 3. This model is particularly useful for businesses that are not certain about their requirements or expect major changes in their mid to high-risk project. Such businesses can benefit from the scalability of spiral methodology.

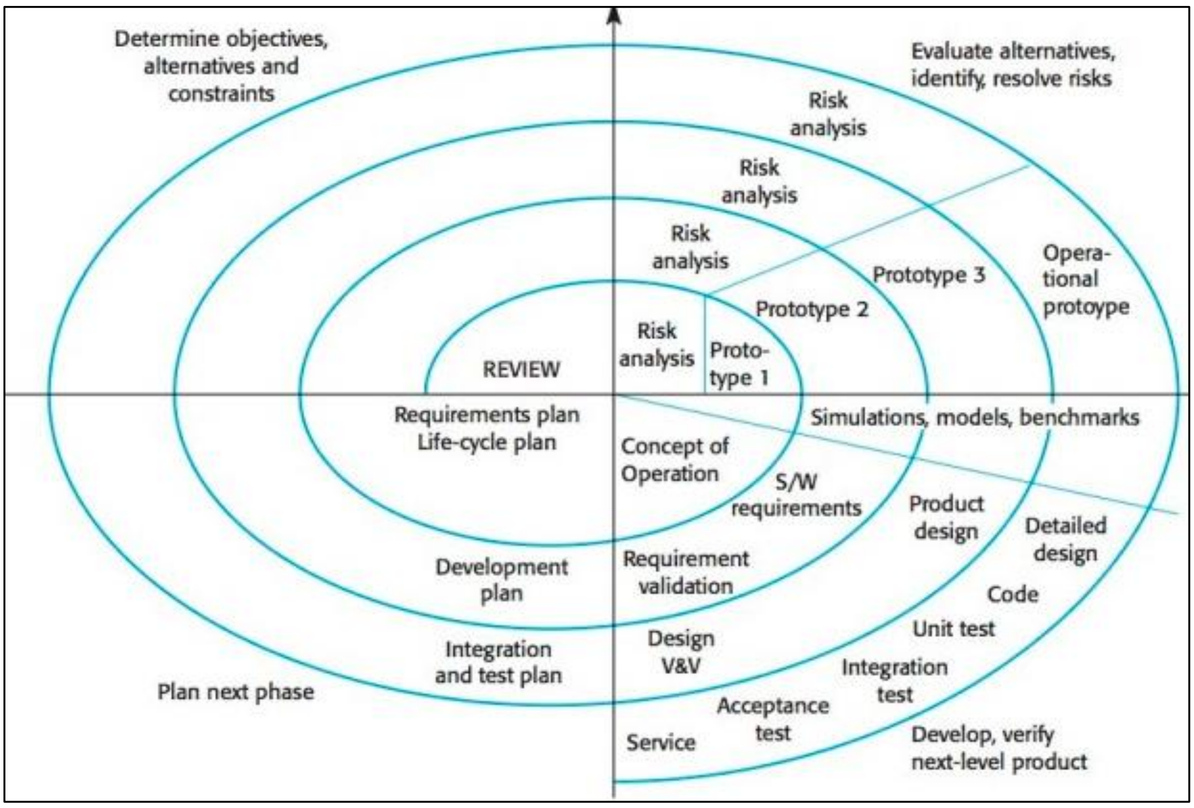


Figure 3: Spiral Methodology (Morley et al., 2000)

2.2.1.3 Prototyping

While the prototyping methodology is focused on producing an early model of an entirely new system, software or application, the prototype would not have full functionality or be thoroughly tested, however, it will provide external customers a sense of what is to come. Afterwards, feedback is generated, and implemented throughout the rest of the SDLC phases as illustrated in Figure 4 below. This methodology works perfect for enterprises in emerging industries or new technologies.

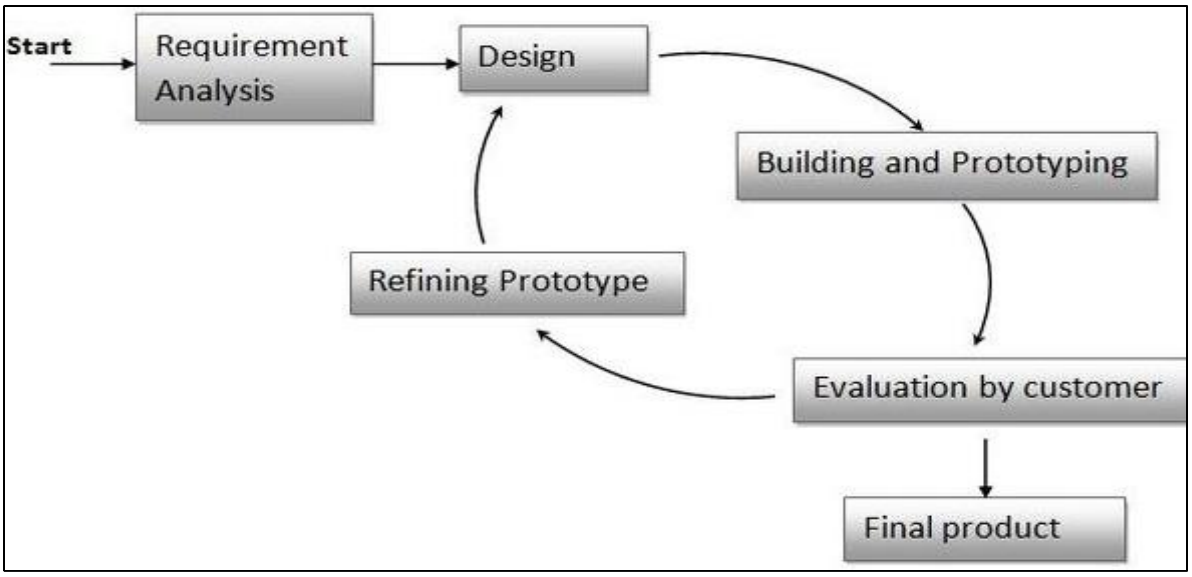


Figure 4: Prototype Methodology (Despa, 2014:45)

2.2.1.4 Rapid Application Development (RAD)

This methodology is a type of iterative and incremental model used to expedite software application development. In general, RAD methodology places more emphasis on adaptive process than on planning. The approach uses predefined prototyping methods and tools to produce minimally-coded software applications. Besides, the core of RAD is prototyping (i.e. creating predefined components, structures and methods to rapidly develop software models). In RAD, the functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. Figure 5 below show the process involved in developing prototypes. The RAD methodology allows rapid development of software from design right through to implementation.

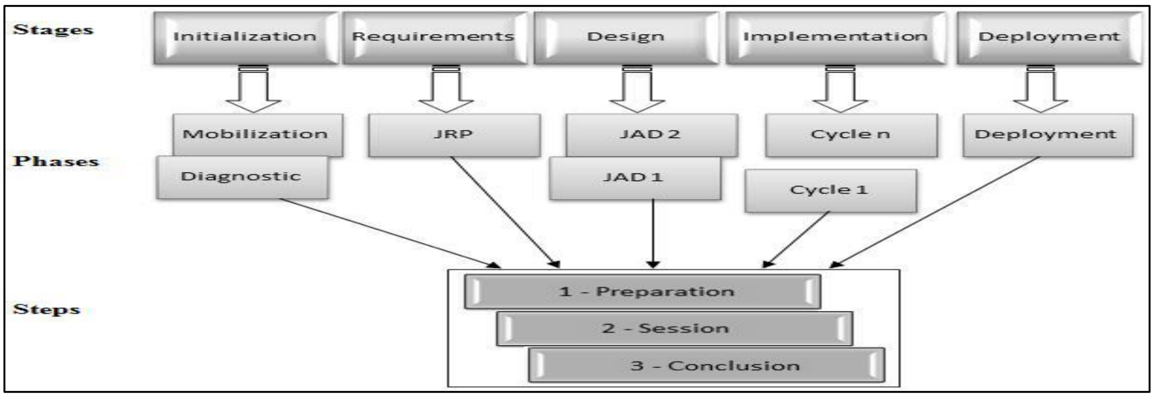


Figure 5: RAD methodology (Morley et al., 2000:131)

While RAD prototypes may lack full-scale functionality, they are useful for demonstration and requirement gathering, and helpful for end users to envisage entire solution stacks. Moreover, because RAD is model-driven and employs an object-oriented approach to developing complete solutions it is most suited for developing software driven by user interface requirements. In addition, the software is grouped into smaller units, this enables changes to be implemented easier throughout the development process. RAD employs other methods of the same nature such as JAD, spiral and prototyping. In RAD, screens displayed during prototyping become the actual screens of the end product software. More so, for this model to be considered successful, it is important that prototypes developed are reusable. The shortcoming of implementing this methodology is that it is possible to miss significant requirements as the team is working through various project iterations (Geambaşu *et al.*, 2011).

2.2.1.5 V-model

The methodology was introduced in the 80s. V-Model methodology is an extension of the waterfall model, and was developed in order to introduce testing throughout the process. It follows a vertical development process with a bottom-up approach for implementation as depicted in Figure 6 (Matthews, 2002). The model presents thorough testing at each software development phase. The left part of the model describes the requirements and design, whereas the right part describes the development and testing activities. The final activity describes the maintenance and support of the end product.

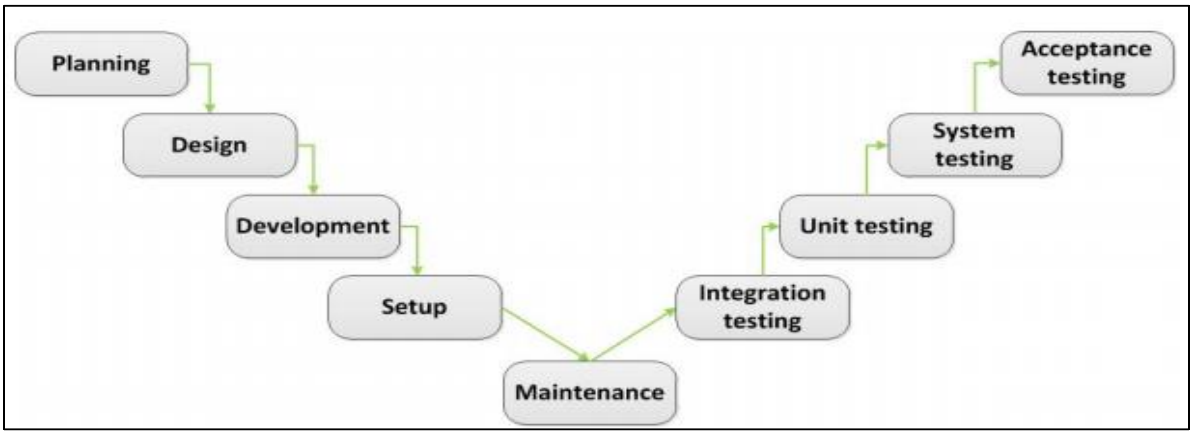


Figure 6: V Model (Despa, 2014:45)

The V-model presents a comprehensive manner of testing as illustrated in Figure 6. Each phase begins with testing. The implication of this is that a good quality integration of the software is provided. Each test is documented and compared with the documents retrieved from the activity, besides the connection through the lines gives a clear overview of the results. Though, similar problems like the one observed using the Waterfall methodology can develop with the V-model method, especially during the start of the process where errors or mistakes in interpretation of requirements and design often happens (Sommerville, 2016). Unfortunately, the V-model is unable to cope with these problems (Aughenbaugh and Paredis 2008). After the entire application is completed, the project owner's feedback is expected as a confirmation of user acceptance testing. The V-Model methodology can be applied to both small and medium scale projects.

2.2.1.6 Problems of the Traditional Methodologies

According to Nandhakumar and Avison (1999:3) traditional methodologies are treated primarily as a necessary fiction to present an image of control or to provide a symbolic status. The authors argued further that these methodologies are too mechanistic to be in daily use by organizations. Parnas and Clements (1986) indicated the same views. Truex et al. (2000:53) was extremely firm in their assertions when they questioned the purpose of these methodologies as to whether they function as frameworks or if the methods are merely unattainable ideals and hypothetical straw men that provide normative guidance to utopian development situations. Therefore, several problems became associated with the use of these methodologies, by more and more IT organizations. Amongst the problems identified were:

- Too much rigidity, i.e., not enough flexibility in general;
- Too heavy;

- Too much linearity, i.e., too much of having to do things in a certain order;
- Too restrictive in what individuals were allowed to do;
- Not enough good communication lines between stakeholders;
- Not adequately utilising resources such as people and time;
- Long development cycles;
- Miscommunication; and
- Bug prone software.

The aforementioned problems provide motivation for the introduction of agile software methodologies. Software development industries have turned to using agile methodologies to manage projects, because unlike traditional approach, agile provide light approaches to projects (Erickson *et al.*, 2005). Moreover, they focus more on managing and speeding up development activities (Goodpasture, 2010). These methodologies are seen as lightweight, and are light in documentation requirements, adaptable to change, and customer focused.

Table 1: Providing an overview of Agile and traditional methods (Leau et al., 2012)

Criteria	Agile	Traditional
User requirements	Iterative acquisition	Detailed user requirements are well defined before coding
Rework cost	Low	High
Development Direction	Readily changeable	Fixed
Testing	On every iteration	After coding phase
Customer Involvement	High	Low
Extra quality required for developers	Interpersonal skills and basic business knowledge	Nothing in particular
Suitable project scale	Low to medium scaled	Large-scale

2.2.2 Agile Methodologies

Agile Development covers several iterative and incremental software development methodologies. Abrahamsson *et al.* (2017b) states that a methodology can only be considered to be agile when it possesses the following features, first, the software development has to be incremental in nature (small software releases, with rapid cycles), second, it has to be cooperative (stakeholders working constantly together with close communication), thirdly, the software development must be straightforward (easy to understand and properly documented), and lastly, it has to be adaptive (flexible enough to make last moment changes).

The most common agile methodologies are Extreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM), Lean Development, and Feature-Driven Development (FDD). Compare to traditional processes, agile processes are more people orientated (Dybå & Dingsøyr, 2008), they do not follow stringent processes, progress is dependable on project group understanding and development; and they promote quicker delivery of working software, they also promote the involvement of the stakeholders (Despa, 2014).

Erickson *et al.* (2005) describes agile processes as stripping off the heaviness of traditional processes in order to encourage rapid responsiveness to changes in the user requirements, timelines or project environment (Erickson *et al.*, 2005). Likewise, Highsmith and Cockburn (2001:122) noted that the significance of agile methodologies lies in the acknowledgement of stakeholders being the primary drivers of project success, effectiveness and manoeuvrability and not necessarily the practices they employ. This summarises the core values and principles that define an agile world view as submitted by Dybå and Dingsøyr (2008), who stated that agile methodology has four central values:

- Emphasize more on individuals thoughts and interactions;
- Less documentations more delivering of operational and tested software;
- Customer collaboration rather than contract negotiation; and
- Responding to change instead of following a plan.

Agile methods are mostly suitable in small organizations, especially start-ups, where speed and flexibility is essential. In this regards, agile SDLC focuses on process adaptability and customer satisfaction. Likewise, it advocates early delivery and encourage rapid and flexible response to change.

Cohen *et al.* (2004) outlines four prescriptive characteristics of agile methods which govern and influence methodology selection. These characteristics are infused in Table 2:

Table 2: Prescriptive Characteristics of some Agile Methods (Cohen et al., 2004)

Agile Methods	Team Size	Iteration Length	Distributed Support	System Criticality
Extreme Programming (XP)	2-10	2 weeks	No	Adaptable
Crystal Methods	1-7	4 weeks	Adaptable	Adaptable
Feature Driven Development (FDD)	Variable	< 4 months	Yes	All types
Lean Development (LD	Variable	< 2 weeks	Adaptable	Adaptable
Dynamic Systems Development Methodology (DSDM)	Not applicable			
Agile Modelling (AM)				

While, team size, iteration length, distribution support and system criticality are instrumental in determining methodology selection, Laurie Williams (2007) additionally noted that because Agile methodology focuses less on documentation, minimal documentation is required which is sufficient enough for the development of software that meets the user's needs. The most effective method for transmitting information to and within the development team therefore is through live communication, instead of written communication. Because it is beyond the scope of this study to fully analyse theoretical frameworks of Agile methodologies, only XP and Scrum will be briefly discussed while a table summarizing characteristics of both traditional and agile software development methodologies will be displayed. This is relevant in order to identify the weaknesses of these methodologies for the purpose of proposing a decision framework for IT organizations to assist in the selection of a development methodology to adopt for a particular project. XP, Scrum and FDD as examples of Agile methodologies that will be analysed, particularly because they are in used in South Africa (Ramnath, 2010).

2.2.2.1 Extreme Programming (XP)

Extreme Programming (XP) like any other examples of agile software development methodologies differ from traditional methodologies because of its ability to emphasise adaptability rather than predictability. Beck and Gamma (2000), describes XP as a software methodology that categorizes users to implement quality software in a fast paced manner. It possesses the characteristics of Agile methodology because it breaks the conventional software development process into several short development cycles. In this regard, according to Beck and Gamma (2000), XP attempts to minimise the costs associated with changing requirements during the system development life cycle.

Furthermore, XP promotes practices found to be effective in software development processes in the past decades (Geambaşu *et al.*, 2011). Among such practice is collective code ownership, whereby, any developer is able to make updates or changes to code packages that were not even written by him (Despa, 2014). These practices disclose two main assumptions - close physical proximity and close customer involvement. According to Geambaşu *et al.* (2011:485) these assumptions increases solid communication amongst members of the project team, while also promoting client interaction throughout the life cycle.

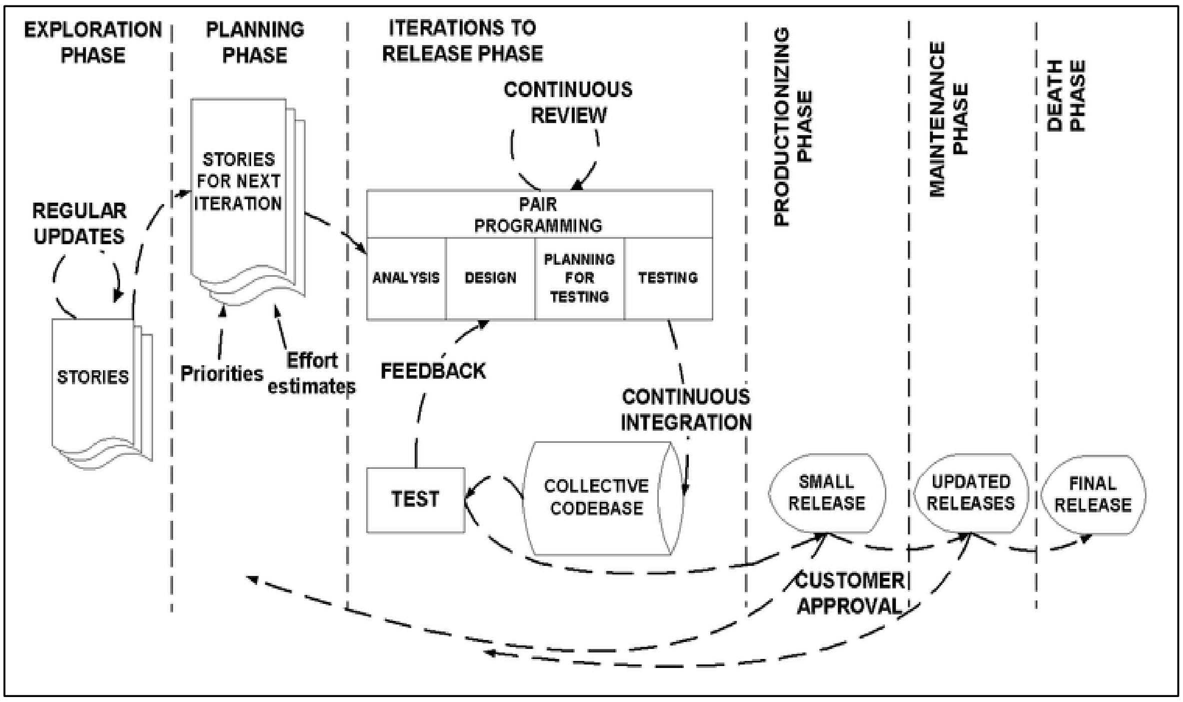


Figure 7: Life Cycle of the XP process (Abrahamsson et al., 2017b: 19)

Figure 7 above illustrates the lifecycle of an XP process. In order to reduce mistakes, XP promotes vigorous unit testing. The software developers are required to produce test cases before compiling the actual software code. In XP methodology, five core values are promoted, which include:

Simplicity: XP entails developers to seek the simplest solutions necessary to satisfy current customer needs while at the same time, they are discouraged from following solutions that solve future problems (Laurie Williams, 2007).

Communication: this is a crucial medium for the exchange of rapid, continual feedback in XP teams. Likewise, it supports agility and the spread of tactile knowledge and permits the team to respond to changing requirements as the client begin to develop a better understanding of the system (Loftus & Ratcliffe, 2005 & Turk et al., 2014). XP promotes face-to-face communication between all stake holders (team members, and client) (Loftus & Ratcliffe, 2005).

Feedback: regular interval feedback value of XP team is critical to the delivery of working software (Turk et al., 2014).

Respect: respect is crucial for XP team members, teams respect each other's expertise and are expected to strive to achieve high quality code and design (Turk et al., 2014 & Laurie Williams, 2007).

Courage: courage is the prerequisite to apply the other XP values (Laurie Williams, 2007), e.g. an environment need team members to be in good interpersonal communication, practicing simple design or generating feedback, and individuals needs to have courage to begin implementing these XP practices. Beck and Gamma argue that courage is important for a team especially when faced with the following situations:

- To make architectural corrections;
- To throw away tests and code;
- To be transparent, whether favourable or not;
- To deliver complete, quality work in the face of time pressure;
- To never discard essential practices;
- To simplify code at every turn;
- To attack whatever code the team fears most (sic); and
- To take credit only for complete work.

The five core values of XP are supported by thirteen important practices namely below:

Pair programming: developers usually work in pairs at a workstation (Loftus & Ratcliffe, 2005).

Collective code ownership: team owns the code base (Loftus & Ratcliffe, 2005).

Continuous integration: daily integration of Code (Loftus & Ratcliffe, 2005; Turk et al., 2014).

Test-first Development: team develops the system through test case writing followed by implementation code (Loftus & Ratcliffe, 2005).

Sit Together: team works in an open space (Laurie Williams, 2007).

Whole Team: The team must be cross functional. This comprises testers, developers, the client and quality assurance team members (Laurie Williams 2007).

Energized work: the XP teams need to work a 40-hour week. Long periods of overtime are not encouraged as they are counterproductive (Laurie Williams, 2007).

Stories: XP team need to write short statements describing the functionality of the desired product (Laurie Williams, 2007). Likewise, they must estimate the size and prioritize the stories (Laurie Williams, 2007).

Weekly cycle: a weekly progress review meeting is advised to allow customer the opportunity to pick a week’s worth of stories to be implemented (Laurie Williams, 2007). More so, such meetings assist to break the stories down into tasks (Laurie Williams, 2007).

Quarterly cycle: this is the period for choosing larger themes or interrelated stories that will be developed over a quarter (Laurie Williams 2007 & Münch, Armbrust, Kowalczyk & Soto, 2012). The implication of this is that themes permit teams to see the larger picture (Laurie Williams, 2007).

Slack: low priority tasks are dropped when team is behind schedule (Laurie Williams, 2007) & Münch et al., 2012).

Ten-minute build: the whole system with the unit test must be built and run in ten minutes (Laurie Williams, 2007 & Münch et al., 2012).

Incremental design: there is a need for daily investment in the design of the system by the team. (Laurie Williams, 2007 & Münch et al., 2012).

XP methodology can only apply successfully when several conditions are met, such as; smaller teams members, the environment must promote continuous communication and team coordination, and all persons involve must accept methodology practices and principles. The XP methodology is appropriate for all kinds of projects from small to, medium and large scale projects.

2.2.2.2 Scrum methodology

Scrum methods are an empirical approach which applies industrial process control theory to system development. Scrum is involved in this study because of its usage in IT firms in South Africa and because it focuses on self-management and on numerous processes involving shared decision-making. Scrum approach is helpful for reintroducing flexibility, adaptability and productivity into software development methodologies (Schwaber & Beedle, 2002). According to Rising and Janoff (2000), scrum is a software development method that is ideal for gradually developing complex software.

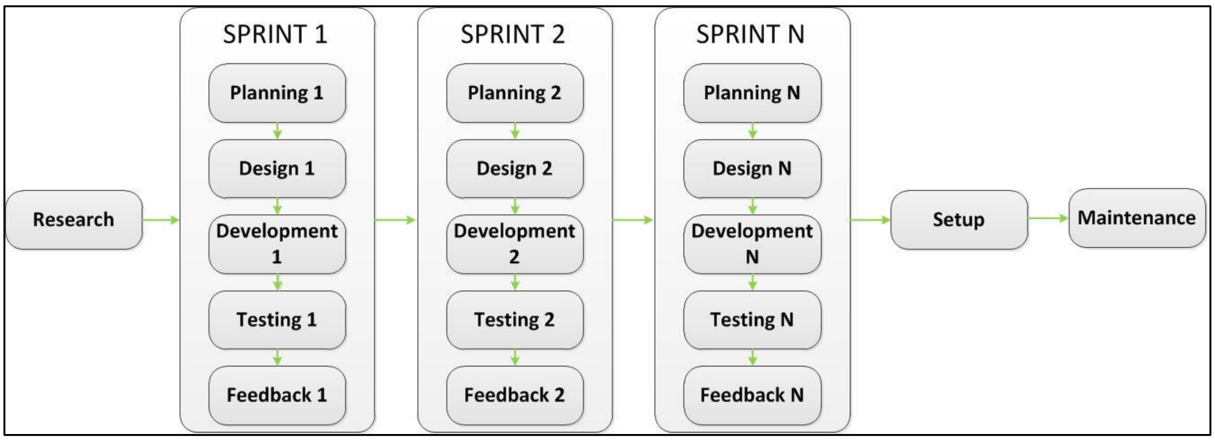


Figure 8: Scrum methodology (Despa, 2014:45)

Moreover, Scrum does not define any specific software development techniques; it concentrates on the functionality of team members so as to produce system flexibly in a continuously changing IT environment. The idea behind Scrum is to meet several environmental and technical variables for example requirements, time frame, resources, and technology, and the ability to be flexible enough to respond to the changes as illustrated in Figure 8 above. Likewise, Scrum assist to advance the existing engineering practices (e.g. testing practices), this is because Scrum involves many management activities targeted at identifying any deficiencies in the development process and the practices that are used (Abrahamsson *et al.*, 2017b). The Scrum methodology can be applied to any project size, small, medium or large.

The Scrum framework consists of the following components:

Roles:

- **Team**: these are the developers in the Scrum team dedicated towards achieving the sprint goal (Laurie Williams, 2007).
- **Scrum Master**: the Scrum is responsible for supervisory roles that assist team resolve issues that are blocking team progress (Lacey, 2012).

- **Product Owner**: the product owner is responsible for the prioritised product backlog which is in the form of user stories (Scharff, 2011). More so, this individual's duties involves making decisions concerning the approval of stories at the end of a sprint (Laurie Williams, 2007).

Ceremonies:

- **Sprint Planning**: in the course of sprint planning meeting, the product owner produces and prioritises the product backlog (Laurie Williams, 2007). Throughout the sprint planning session the team agrees on a sprint goal, which assists as the success criteria for the sprint (Laurie Williams, 2007).
- **Sprint Review**: this provides an opportunity for the team members to show its accomplishments during the sprint (Lacey, 2012).
- **Sprint Retrospect**: retrospectives are significant for the continuous development of the team (Lacey, 2012). Retrospectives also provide opportunity for team members to reflect on how to improve efficiency, quality and velocity (Lacey, 2012).
- **Daily Scrum Meeting**: this refers to 15 minutes daily meetings whereby each team is obligatory to provide answers to the following three questions (Laurie Williams, 2007):
  - What did you do yesterday?
  - What will you do today?
  - What is blocking you from completing your tasks?

Artefacts:

- **Product Backlog:** refers to a prioritized master list of requirements that encompasses the vision of the product to be developed (Williams, 2007 & Lacey, 2012).
- **Sprint Backlog**: refers to a list of tasks needed to be completed by the team during the sprint (Lacey, 2012).
- **Burn down Charts:** a graphical representation of the work remaining (Williams, 2007 & Lacey, 2012).

2.2.2.3 Feature Driven Development (FDD)

FDD emphases on delivering tangible functionality in 2 week iterations and is designed to be used in conjunction with other development activities (Mnkandla, 2009). Feature Driven Development has the following artefacts and roles as depicted in Figure 9:

Artefacts:

- **Feature list**: useful features for the client (Laurie Williams, 2007).
- **Design packages**: describes notes, class diagrams, and sequence diagrams reports (Palmer & Felsing, 2002).
- **Track by feature**: a graph indicating dates when features are to be released (Laurie Williams, 2007).
- "**Burn Up" chart**: refers to a chart showing project scope and work completed (Laurie Williams, 2007).

Roles

- **Project manager**: project administrative leader (Laurie Williams, 2007).
- **Chief architect**: the individual accountable for overall design (Laurie Williams, 2007).
- **Development manager**: The person answerable for daily development activities (Laurie Williams, 2007).
- **Chief programmer**: An experienced the team leader (Laurie Williams 2007).
- **Class owner**: the person accountable for overall designing, testing and documenting features (Laurie Williams, 2007).
- **Domain experts**: expert individual(s) with deep knowledge about the business (Laurie Williams, 2007).
- **Feature teams**: deals with implementing features (Laurie Williams, 2007).

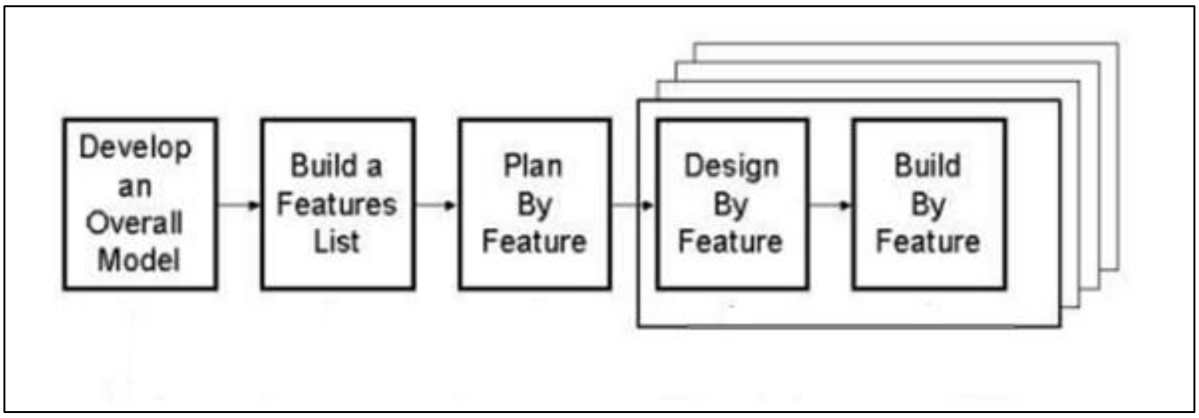


Figure 9: Graphical representation of the Feature Driven Development Model ((Palmer & Felsing, 2002).

2.3. BENEFITS OF AGILE ADOPTION

Past studies (Dybå & Dingsøyr, 2008; Sharp & Robinson, 2004; Poole *et al.*, 2001) have shown an increase in productivity when agile practices were adopted. Besides, the productivity increase mentioned by Poole et al (2001) was also attributed to an increase in moral resulting from paired programming (Poole *et al.*, 2001).

More so, a study at IBM on a small team (7- 11 team members) reported an improvement in productivity with a 40% reduction in pre-release defect density when compared to the same metrics from an earlier release (Layman *et al.*, 2004).

In a case study conducted by Bedoll (2003) on infinite productivity consisting a team of 20 developers using Boeing's standard development methodology, the team reported that an initial release was scrapped after two months of trials. Nevertheless, a second attempt by a two-person team involving the use of practices similar to that of Extreme Programming (EP) delivered a working product in only six weeks.

Most studies (Dybå & Dingsøyr, 2008; Laanti *et al.*, 2011; Murphy *et al.*, 2013; Senapathi & Srinivasan, 2011; Sriram & Mathew, 2012;) submitted positive evidence that Agile methodology have several adoption benefits. These benefits include: better software quality, upgraded quality of the development process, decrease in defects, reduced time to market and documentation, improved customer collaboration, shared learning, developed communication and productivity, better predictability and improved transparency. The benefits are tabled in Table 3.

Table 3: showing the key features/Benefits of Agile Methods (Vanker Cassim, 2015)

Feature	Benefits
Continuous requirements gathering	Provides flexibility by allowing customers to delay crucial decisions
Frequent face-to-face interactions	Building trust and overcoming misunderstanding amongst team members
Pair programming	Improves code ownership and promotes teamwork.

Refactoring	Progressive improvement of code without creating shock waves
Continuous release and integration	Supports early detection and correction of bugs. Resulting in higher quality software.
Early expert customer feedback	Reduces costly code overhauls in the end. Also lowers the cost of development.
Reduced documentation	Lowers the cost of documentation. Resulting in shorter development time.

2.3.1 Problem of agile software methodologies

Agile methodologies offer many benefits to an organization over traditional plan-driven approaches, such as increased customer collaboration, improved time-to-market, productivity and quality software, learning-in-pair programming, and thinking ahead for management (Dybå & Dingsøyr, 2008). There are some shortcomings as well, for example concern has been raise as to adopting agile methodologies in large-scale distributed project development environments (Laanti, 2013). Likewise, Manawadu *et al.* (2013:5) noted that the lean development technique reported some issues for teams trying it out when pair programming was inefficient. XP has been criticised as appropriate with experienced development teams. Moreover, past studies have recognised that Agile methodologies lack attention to design and has architectural issues (Manawadu *et al.*, 2013; Rosenberg & Stephens, 2008).

2.4. Overview Of Strengths And Weaknesses Of Traditional And Agile Methodologies

The previous sections have discussed traditional and agile methodologies, while also highlighting various related problems. When developing a decision framework, there is a need to further provide an overview for various characteristics, strengths and weaknesses of these methodologies. This section caters for this.

Table 4 below offers the main difference between the processes of these methodologies, which is based on processes related to projects and their surrounding environments. For example, traditional processes view the world as fully specifiable and the environment is considered stable. On the other hand, agile processes acknowledges that changes always occur, particularly in such a dynamic market. These processes are so intended that element specification takes place at a later stage of the process, in which more knowledge has been gathered concerning the project and the environment.

The implication of this is that there is a total different approach to the organization and management of a project and project team. Traditional projects requires tight coordination and promotes individualization, whereby each member of a project team works independently. While traditional methods is predictive, agile is adaptive, and requires a much more flexible project team. It advocates a small and collaborative team that work closely together. Team members are expected to work collaboratively which allows knowledge sharing among team members and business partners.

Table 4: Differences between traditional and agile development processes (Sharon *et al.*, 2010:44)

	Traditional processes	**Agile processes**
Assumptions	Software is specific and predicable and is well planned.	Software is developed following continuous/ iterative approach, based on regular feedback.
Control Style	Follows hierarchy where there is command and control.	High interaction and collaboration.
Content Management	Content is explicit	Static knowledge
Communication	Formal	Casual or informal
Ideal organizational structure	Large formal cooperation with some level of bureaucracy.	Flexible, cooperative and social action
Quality Assurance	Vigorous planning and stringent controls	Ongoing control

The differences between the software processes and categories highlighted in Table 4 are also found in the characteristics, strengths and weakness of each individual process as presented in Table 5 below. The information displayed in Table 5 is grounded on literature review as mentioned in the previous sections.

Table 5: Characteristics, Strengths and Weaknesses of some of the common software methodologies (Despa,

2014: 51-54)

	Characteristics	Strengths	Weaknesses
Waterfall	Proper documentation is required. Proper planning is required. This methodology follows a linear or sequential process. Each stage has its own deliverables.	Clear, easy to understand and easy to manage for the user and software team.	Software is delivered at the end of the project. Cannot handle changing requirements. Low tolerance for planning and design errors.
Prototyping	Stakeholders are actively involved. A demo version is developed. writing code is valued over writing specifications	Accurate understanding of application requirements. Regular feedback from project owner. Emphasis on user experience. Early identification of errors in functionality.	Often leads to increase of application complexity. Costs of generating a demo version/prototype
Spiral	Consists of four main phases: Planning, risk analysis, implementation and evaluation/testing. This methods emphasis on risk analysis. Various options are evaluated before proceeding to the planning phase.	Risk is minimized. Documentation is well maintained. Working code is delivered early in the project.	Depends heavily on risk analysis. The costs of risk mitigation/handling may be high.

	Characteristics	Strengths	Weaknesses
Rapid application Development	Follows a time box approach. Effort and emphasis is given to development rather than planning tasks.	Software is produced rapidly, and the code is easily reusable.	Documentation is not well maintained. Development costs are high. Application is broken down into modules, therefore integration issues exist.
V-models	Testing happens after every development stage. Maintenance of software is emphasized.	Simple to understand. Less errors or bugs due to vigorous testing.	Prone to scope creep. Follows initial set of requirements. Not flexible to change.
Scrum	Follows agile approach. Iterative development style. Groups' development tasks into sprints. Daily feedback meetings. Organised teams and managed tasks.	Quicker development cycles. Regular feedback. Adaptable to change.	Poor documentation. Requires experienced developers. Poor effort estimation and cost estimations.
	Puts effort into determining the requirements. Involves the stakeholders and end users. Requirements are jointly developed in JAD meetings.	Quicker design. Clear requirements. Promotes teams work. Customer centric.	Dependant on successful JAD meetings. Poor documentation after system design phase.

2.5. Project Characteristics and Adoption of Software Development Methodologies

Mahapatra (2015), describes software development methods (SDM) as a standardised approach to the implementation of software solutions. The main purpose of his research paper was to discuss and compare the different methodologies in order to be able to choose the most suitable methodology for a specific project. Munassar and Govardhan (2010) purported to present different models of software development and draw a comparison between them. In their study, the authors focused more on important issues in the computer world and concerned themselves with examining the various software development methodologies. Five different methodologies were taken into consideration, namely, waterfall, Iteration, V-shaped, Spiral and Extreme Programming, mainly to show the features and defects of each model. The study however, did not discuss what make these methodologies appropriate for use.

In addressing the above gap, Despa (2014:55) asserted that in choosing an appropriate software methodology, certain factors will need to be taken into consideration. Factors such as project stakeholders, developer's skills, project complexity, costs and timelines. The author confirms that no single method will perfectly suited to the profile of a specific project. However, Despa (2014) stressed that the best matching methodology should be used in an organization. More so, a combination of methodologies may be introduce in cases of experienced project teams and project managers while a new methodology is advised for innovative software development projects.

Abrahamsson *et al.* (2017b), claimed that the introduction of several different approaches to software development in the past 25 years is not the solution to achieving success in software development. These authors reviewed and analysed agile software development methodologies for the purpose of enabling software professionals, projects and organizations to choose and adopt the best software methodology, and their finding begged for an urgent need for the adoption or selection frameworks to be used by practitioners rather than introduction of new methodologies.

Meanwhile in New Zealand, Sheffield and Lemétayer (2013) explored factors associated with the software development agility of successful projects, using qualitative methodology to determine what factors in the project and its environment may be suggestive of software development agility in successful projects. The study findings showed that software development agility was indicated by a project environment factor (organizational culture) and a project factor (empowerment of the project team). While the findings may assist practitioners in reflecting on development practices and to negotiate change towards achieving higher rates of project success, the study only focused on organizational culture and empowerment of project teams, whereas project characteristics may be crucial in choosing software methodologies. In an investigative study, Vijayasarathy and Butler (2016), tested whether organizational, team and project characteristic matter when choosing software methodologies. Project managers and other team members were surveyed about their choice of methodologies through an anonymous online survey. Study findings indicated that while agile methodologies such as Agile Unified Process and Scrum have been dominant over 10 years ago, traditional methodologies, like the waterfall model, are still popular and in use today. Likewise, organizations are taking a hybrid approach, using multiple methodologies on projects. Besides, their choice of methodologies is connected with certain organizational, project, and team characteristics.

Although, organization, project and team characteristics may be relevant, a decision support framework remain significant in the selection for appropriate methodologies as acknowledged recently by Abrahamsson, et al., (2017:106), who stated that introduction of several software methodologies is not the solution to addressing software crisis, but rather having a decision framework in place. According to Abrahamsson, et al., (2017:106) the frequent release of new agile methods into the market will bring about confusion instead of clarity. This is because each method uses different vocabulary and terminology. In the process of integrating these many viewpoints, minimal work is achieved. Therefore an evaluation of project characteristics becomes critical.

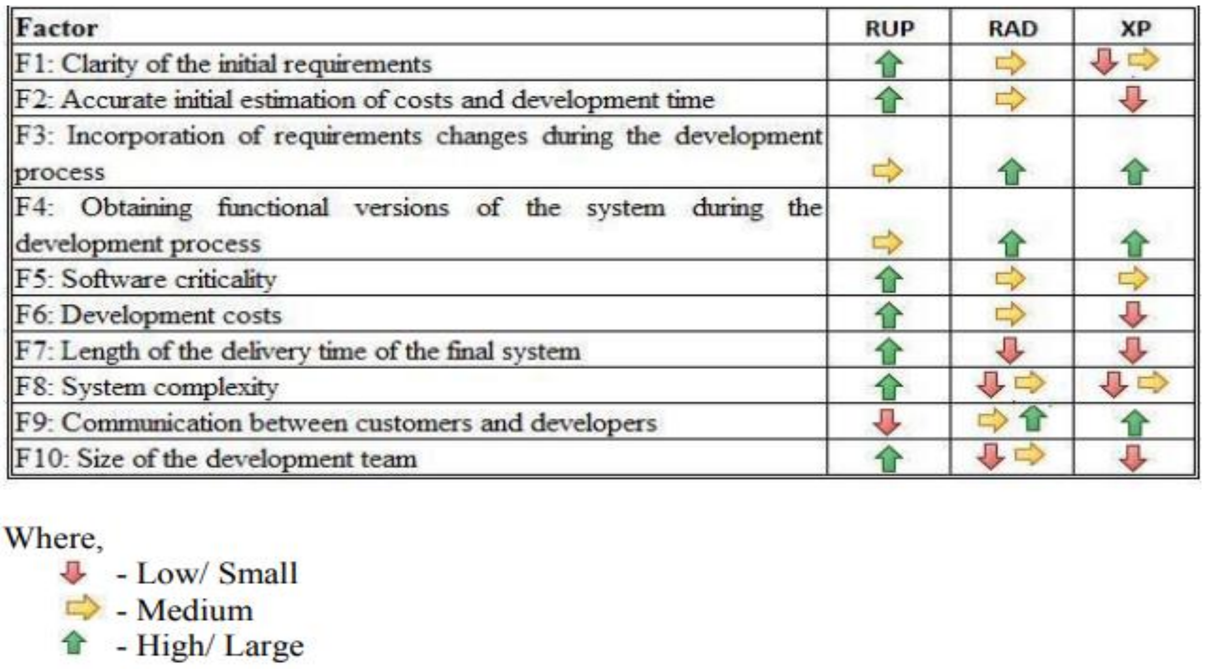
In the United States, Harb *et al.* (2015) evaluated project characteristics for the Best-fit Agile Software Development Methodology. This is because they acknowledged that choosing an appropriate software methodology is complex, which requires a multi-criteria decision approach and this have implications for project success. Harb *et al.* (2015) offered solution by presenting a teaching case designed especially to support Information Systems students, improve skills in understanding and evaluating complex business requirements, and help in selecting the most appropriate software development methodology vis-a-vis needs of a specific IT project, and the organization. In an Evaluation study of software development methodology adoption in Sri Lanka, Manawadu *et al.* (2013:8) found that choosing an appropriate software methodology was due to the nature of project management.

The views of early scholars are different with regard to factors that influence selection of the most appropriate software development methodology. For example, grounded on the investigation of over one hundred software organizations, Russo *et al.* (1995) found that “the three most important features both for selecting and using the methodologies were: structured development techniques, well-defined corporate policies/procedures, and sharing of information between developers”. Cockburn (2000) recognizes only two factors, namely: project priorities and the methodology designer’s peculiarities. Yet, the shortcoming in these studies are their inability to analyse and identify specific development methodologies relative to these factors.

In analysing key factors that influence choosing the most adequate software development methodology, Geambaşu *et al.* (2011:491) analysed RUP, XP and RAD methodologies, unlike the scholars named above, Geambaşu *et al.* (2011:491) identified ten factors that influence the decision of choosing the most adequate development methodology for a specific project (see identified factors in Table 6 below).

Table 6: The level of factors for which the software development methodology is appropriate (Geambaşu et al.

(2011:490))



Factor	RUP	RAD	XP
F1: Clarity of the initial requirements	⬆	➡	⬇ ➡
F2: Accurate initial estimation of costs and development time	⬆	➡	⬇
F3: Incorporation of requirements changes during the development process	➡	⬆	⬆
F4: Obtaining functional versions of the system during the development process	➡	⬆	⬆
F5: Software criticality	⬆	➡	➡
F6: Development costs	⬆	➡	⬇
F7: Length of the delivery time of the final system	⬆	⬇	⬇
F8: System complexity	⬆	⬇ ➡	⬇ ➡
F9: Communication between customers and developers	⬇	➡ ⬆	⬆
F10: Size of the development team	⬆	⬇ ➡	⬇

Where,

⬇ - Low/ Small

➡ - Medium

⬆ - High/ Large

Replicating the findings of Harb *et al.* (2015) in another study in the United States, Kamal (2015) employed an action research methodology to investigate the extent to which Information Technology adoption and usage could be sustained in micro-enterprises so as to facilitate business growth. The study developed an online tool to facilitate micro-enterprises' sustainability of ICT adoption and use. This study was however limited as it did not provide a framework for the adoption of software development methodologies.

In closing the above gap, Khan and Beg (2013) investigated utilising the extended decision support matrix for selection of SDLC models on traditional and agile software development projects. The authors acknowledge the risks associated with wrong selection of SDLC-models on business critical software projects and offer a pragmatic solution by proposing a handy selection matrix for choosing the best SDLC models on different types of Software Development Projects. However, this research only focuses on traditional and agile development projects, but did not consider the level of adoption and effectiveness of software development methodologies.

Ramnath (2010) examined the extent to which software development methodologies are effective in the software industry in South Africa. In an attempt to establish whether the factors influencing the selection are indeed realised Ramnath (2010) found that the agile methodology is currently the most prevalent in the South African software industry, and is furthermore the most preferred by professionals in this field. Moe *et al.* (2012) argue that preferring agile development should not alter the important knowledge required in software development however, it only changes the manner of coordination and collaboration in software projects. According to Moe *et al.* (2012) this may intensify development problems (Dybå & Dingsøyr, 2008). Therefore, determining decision making in this context becomes imperative.

Moving from a traditional methodology to an agile approach may be a part of organisation's strategy (Munassar & Govardhan, 2010), but it is important that professionals are clear about their role in fulfilling this strategy. A failure of such recognition may easily predict a failure of the agile strategy itself (Delcheva, 2018:13). Giving an holistic overview Delcheva (2018:6), stated that most organisations, transiting to Agile from traditional approach, overlook the different backgrounds of both approaches with their different principles (Fitzgerald, 1996). Stressing further that the majority of literature on agile development focuses more on the comparison between these two methodologies, while in effect the project manager role is also important in a successful business strategy.

The assumption of aligning product and project decisions with business strategy become important for the successful application of an agile approach in software development, and this was the motivation for a study on challenges of shared decision-making by Moe *et al.* (2012:854). In understanding the challenges of shared decision-making in agile software development teams among two software product companies, Moe *et al.* (2012) designed a multiple case study involving three projects that recently implemented Scrum. Results identified three major challenges associated with shared decision-making in agile software development, i) alignment of strategic product plans to iteration plans, ii) development of resource allocation, iii) performing development and maintenance tasks in teams. The limitation of this study is that it only focused on software product companies using Scrum approach.

Bassil (2012) studied the waterfall model, an example of Software development life cycle (SDLC), because SDLC is continuously plagued by budget overrun, late or postponed deliveries, and disappointed customers (Leung & Fan, 2002). A situation confirmed by the Standish Group (2015), who reported that several projects were unsuccessful based on delivery time, budget and requirements. Bassil (2012) attached these failures to project managers, stressing that project managers are not intelligently assigning the required number of employees and resources to the various activities of the Software Development Life Cycle.

This will explain why some SDLC phases are been delayed, with other dependent phases staying idle, until completion of other phases, resulting to a bottleneck between the arrival and delivery of projects which leads to a failure in delivering a functional product on time, within budget, and to an agreed level of quality. To solve this situation, Bassil (2012) proposed simulation for the Waterfall model. Although this was meant to address the issue of trade-offs between cost, schedule, and functionality. However, this study was not conclusive in that a framework to guide the adoption of software development methodologies was not formulated.

SDLC of software systems has always encountered problems and limitations that resulted in significant budget overruns, late or suspended deliveries, and dissatisfied clients

In a mixed method study in Brazil, Roses *et al.* (2016) proposed and tested a model to evaluate the degree of conditions favourability in the adoption of agile methods to develop software particularly where traditional methods predominate. Using surveying methods and applying factorial and frequency statistical analyses on quantitative data and thematic content analysis to analyse qualitative data, a model was proposed to examine the degree of favourability conditions in the adoption of agile practices within the context of software developers within the banking field. This study limitation was its focus on only traditional software methodologies.

To close the above gap, from a vendor's perspective, Rajagopalan and Mathew (2016) assessed how Vendor firms made choices on agile methodologies in software projects as well as their fit. This resulted in the development of two analytical frameworks from literature. Findings were compared with real life decisions. Findings on Framework 1 showed that the choice of XP for one project was not a supported base for framework guidelines. While the choices of Scrum for other two projects, were partially supported. Analysis using the framework 2 revealed that with the exception of one XP project, the rest had adequate project management support, limited scope for adaptability and had prominence for rules.

Moniruzzaman and Hossain (2013) conducted a comparative study on agile software development methodologies and they identified that Agile development approach improves software development process so as to meet the rapid changing business environments. A brief comparison of both agile and traditional development methodologies were discussed. According to Moniruzzaman and Hossain (2013), agile software development emerged as a substitute to traditional software development methods, this is to satisfy customers through timely and continuous delivery of the valuable software.

2.6. ORGANIZATIONAL FACTORS INFLUENCING THE CHOICE OF SOFTWARE DEVELOPMENT METHODOLOGIES

It is important to identify the underlying factors to be considered when adopting a software development methodology. Geambaşu et al. (2011) identified and analysed the key factors that influence the selection of the most appropriate software development methodology for a specific project. The researcher further analyses some common software development methodologies i.e. RUP, XP and RAD with the purpose of finding the relationship between the key factors affecting the model.

Cockburn (2000) identified two factors that affect the methodology selection, namely, the project priorities and the designer's peculiarities. However, both researchers did not analyse the link between these factors and the actual software development methodologies, for instance, which methodology should be applied in accordance to the identified factors.

The software development life cycle adheres to the basic rules of project management, although includes some added features. A software project manager is required to manage arising issues that are propriety to the software industry. Also, in software development there are some benefits that make it easy to manage the software project.

Liviu Despa (2014) addresses the current state of software methodologies. The aim was to standardise the software development methodologies that are dedicated to innovation and information technology. The author begins by presenting specific characteristics in software development projects. The methodologies are then compared by discussing their strengths and weaknesses from the user's perspective. Conclusions were formulated and a formalised methodology is enunciated.

Based on factors (e.g. high budget, improper schedule, and failure to meet customer expectations) that characterizes failures of software projects for developers, Kumaresan and Kumar (2018) presented factors and methods for software project success. Factors such as organizational, technical, people, and cultural factors were identified. Situational factors however may be critical in predicting the choice of software development methodologies

Clarke and O’Connor (2012) investigated situational factors affecting the software development process Factors such as the nature of the application(s) under development, team size, requirements volatility and personnel experience were acknowledged to be factors affecting software the development process, yet there is a lack of reference framework addressing situational factors affecting the software development process. This is problematic because it inhibits the ability to optimise the software development process, and potentially undermines the capacity to determine vital limitations and characteristics of a software development setting.

To address this deficiency, Clarke and O’Connor (2012) consolidated a substantial body of related research into an initial reference framework of the situational factors affecting the software development process and found eight classifications and 44 factors that inform the software process. The situational factor reference framework presented only represents the key situational elements affecting software process definition while organizational factors were ignored.

Ezeh and Anthony (2013) explored organisational factors that have positive and significant impact on knowledge sharing. By means of a case study carried out at Volvo Cars IT (VCIT), Torslanda, software development professionals were to identify different perspectives on organisational factors that influence knowledge sharing. Through thematic analysis, results indicated the factors as social relations and network, physical closeness to colleagues, no stupid question culture, mutual exchange, interest in work involvement, satisfaction of helping each other, being listened to and taken seriously, and satisfaction from personal goal. The limitation of this study is that only the organisational factors influencing knowledge sharing was identified whereas organizational culture may be vital in software development process.

In a cross-case analysis across four software organizations in Brazil, Passos *et al.* (2014) investigated the role of organizational culture in software development organizations. Through employing the Theory of Reasoned Action (TRA) they analysed the connection to origins, sources and impacts of beliefs on software development practices. The authors provided narrative accounts of software project teams, relating the influence factors associated to team belief systems and attitudes toward practice. The results presented strong influence of past experiences and organizational contexts on software development practices. The short coming of the study was that it was conducted in Brazil only.

To close the above gap, Khoza and Pretorius (2017) examined negative factors influencing knowledge sharing in software development in South Africa. The study wasas conducted using expert sampling to derive data from software development projects team members. Four Johannesburg-based software companies participated in their study; the findings of this study provide some compelling insights. It was revealed that job security, motivation, time, physiological factors, communication, change and rewards are core factors which negatively influence knowledge sharing within software development organisations. In conclusion, the authors highlighted the importance of understanding the negative factors in order to assist software development organisations to close the gap and ensure that software projects are delivered in time, within budget and within scope. However, the limitation of this study is its inability to identify organizational characteristics and structures.

In identifying factors influencing software development methodology, Farrell (2007) evaluated each software methodology based on organizational characteristics. Organizational characteristics and structures were examined. Characteristics associated with organizational bureaucracies, namely, organizational structure, software complexity, effort, work type, change management, and organizational size were discussed and analysed. According to Farrell (2007) the type of organizational structure will predict the most suitable software development methodology to adopt in that organization. Apart from organization structure, organisation culture and top management support may also be crucial factors influencing the choice of software development methodologies.

In Taiwan, Lee *et al.* (2016) investigated the impact of organizational culture and management support on the success of software process improvement (SPI). An innovative model was developed to test the influence of knowledge sharing on SPI success, the effect of knowledge sharing in specific organizational cultures, and the extent to which the support of top management have influence on the path to SPI success. Using partial least squares (PLS) statistics to analyse 118 samples from Taiwanese SPI organizations. Findings suggest that that clan-type organizational culture has a stronger relationships with knowledge sharing than hierarchy-type culture in the context of SPI success. SPI knowledge sharing is indicated to be a mediator between clan culture and top management support within the context of SPI success.

Colomo-Palacios *et al.* (2014) investigated the implications of Global Software Development (GSD) for software project managers by analysing project performance from several perspectives (e.g. the 360-degree feedback evaluation). Findings suggests that performance of

GSD projects is lower than in-house projects, further findings shows negative consequences for software project managers, that needs serious consideration. For example, the experiment indicated inattention to tasks by software project managers that led to performance losses. People factors may be crucial factors influencing the choice of software development methodologies.

Lalsing *et al.* (2012) examined the fundamental people factors to consider, in order for a team to be effective, when adopting Agile. Using objective measures, subjective measures and a survey findings suggests that for Agile methodologies to work well, it is crucial to choose the right people for the right team. However, for Highsmith and Highsmith (2002:102), organisation culture is the most crucial factor and they have highlighted the importance of aligning the methodology to the organisational culture. Organisations that promote competence and collaboration are more suited for agile methods, rather than those relying on more vigorous control and planning. There was no concrete support provided to substantiate these views.

2.7. SUMMARY OF THE CHAPTER

The chapter reviewed existing research on traditional and agile software methodologies, with special emphasis on characteristics, strengths and weaknesses of these methodologies, which was intentional to identify the need for developing a decision support framework. In addition, it discussed in detail project characteristics and organizational factors influencing the selection of software development methodologies. In summary, previous studies have only focused on the below:

1. Traditional and agile development projects;
2. Factors affecting the choice of Software Life Cycle Model;
3. Comparative study of various SDLC models; and
4. Challenges related to the adoption of various SDLC methods.

This study is different from the other studies in that the outcome of the study is to formulate a framework that will guide professionals within the IT industry in selecting the most appropriate methodology to adopt in their software development projects. This will reduce the problem of failing projects. An overview summary of literature review is available in Table 7 below.

Table 7: Overview of Literature Review

Author	Research Topic	Research Methodology	Result
(Khan & Beg, 2013)	Extended decision support matrix for selection of SDLC models on traditional and agile software development projects.	The methodology adopted in this study was a combination of exploratory research as well and grounded theory research methods. (Survey was used to implement these methodologies).	This researcher only focuses on traditional and agile development projects; this study did not provide a framework or a guide for the adoption of software development methodology for various projects.
(Ramnath, 2010)	The extent to effectiveness and adoption of software methodologies in South Africa	This study is quantitative in nature and follows descriptive research approach to respond to research questions within the South African context.	This study found that the Agile method is most prevalent and preferred methodology used currently and to be used going forward in South Africa. However this study was not conclusive in that a framework to guide adoption of software development methodologies was not formulated.
(Mahanti et al., 2012)	An Empirical Study of Factors Affecting the selection of Software Life Cycle Models in the Software Industry.	The methodology used in this study is a survey-based approach.	This study revealed that the level of understanding of the requirements is the most crucial factor in the choice of the model used in the software project. Project Complexity is the second leading factor. Man-machine Interaction is the least important factor in the choice of software development methodology; however no guide or framework was implemented for the

Author	Research Topic	Research Methodology	Result
			adoption of software development methodologies. This study was conducted in India and not in South Africa.
(Alashqur, 2016)	Towards A Broader Adoption of Agile Software Development Methods.	The methodology used in this study is a quantitative based approach.	In this paper, an analysis is provided of several practices and techniques that are part of agile methods that may hinder their broader acceptance. Further, solutions are proposed to improve such practices and consequently facilitate a wider adoption rate of agile methods in software development. However no guide or framework was implemented for the adoption of software development methodologies.
(Sharma & Singh, 2015)	Comparative Study of Various SDLC Models on Different Parameters.	The methodology used in this study is a quantitative based approach.	In this paper a comparative study of the following software models namely Waterfall, Prototype, RAD (Rapid Application Development) Incremental, Spiral, Build and Fix and V-shaped was performed. The main objective of this research is to represent different models of software development and make a comparison between them to show the features of each model. However this paper does not provide a guide or framework for the adoption of software

Author	Research Topic	Research Methodology	Result
			development methodologies.
(Campanelli & Parreiras, 2012)	A Conceptual Model for Agile Practices Adoption.	The methodology used in this study is a questionnaire-based approach.	This study proposed a model to store information about agile principles, methods, practices and correlated information to allow organizations to understand their current practices versus agile practices and define the strategy of agile practices adoption to achieve the organization's goals. However this study does not provide a guide or framework for the adoption of software development methodologies.
(Duggal, 2006)	Guidelines to support choice of development methodology.	The main methodology used in this study includes gathering, understanding, analysis and presentation of source material. This was carried out by the literature searching of journals, books, reports and periodicals contained both in the library and on the internet.	The aim of this study was to investigate what factors should be considered when deciding what development methodology to use if any to support development of a particular project. However this study does not provide a guide or framework for the adoption of software development methodologies.
(Floraet al., 2014)	Adopting an Agile Approach for the Development of Mobile	An extensive survey was conducted for this study to gain a better understanding of suitable Agile	Agile processes were considered to be very appropriate with software for fast-paced markets, where customer satisfaction is governed by early and frequent

Author	Research Topic	Research Methodology	Result
	Applications.	approaches currently practiced by mobile companies for the development of mobile applications.	delivery, where there is scope for changes even late in the project, the delivery cycle is short (e.g. every couple of weeks), there is appropriate collaboration between businesses and developers, where working software is the primary measure of progress, and where there is continuous attention to technical excellence and good design and simplicity. This study only focuses on the adoption of Agile approach in the mobile applications development industry.
(Manawadu et al., 2013)	An Evaluation of Software Development Methodology Adoption by Software Developer in Sri Lanka.	This research is an exploratory study. The research adopted a hybrid approach mainly based on quantitative research methodologies which slightly combined with qualitative research methodologies.	The main goal of the research was to understand the current usage of software development methodologies used by the software developers in Sri Lanka. Also to understand how the methodology usage has evolved during the past decade. This study was not conducted in the south African context and it also does not provide any insight in the adoption framework for software development methodologies.
(Kanane, 2014)	Challenges related to the adoption of	The methodology used in this study is a quantitative based	This study displays that the nature of Scrum makes that there is not a single way of adopting it.

Author	Research Topic	Research Methodology	Result
	Scrum	approach. This research was conducted in cooperation with a software development company in the financial technology field.	Adopting Scrum is more a process of continuous adaptation and improvement, therefore facing challenges is an inseparable part of this process. This paper focuses on the agile software development methodology; it does not provide a full analysis of all software development methodologies and does not provide a guide or framework for the adoption of software development methodologies.
(Abrahamsson *et al.*, 2017a)	Agile software development methods: Review and analysis.	The methodology adopted in this study is that of a systematic review approach	The researchers recognized the introduction of several different approaches to software development in the past 25 years, and based on the systematic analysis they conclude an urgent need for the adoption or selection frameworks to be used by practitioners. Rather than introduction of new methodologies.
García, Moraga, Serrano & Piattini, 2015.	Visualisation environment for global software development management	The methodology used in this study is a questionnaire-based approach, all questions were closed-type.	DESGLOSA-GSD was developed with JAVA technology for the visualization of indicators in a Global Software development (GSD) context. This was to support the decision-making process in GSD contexts

Author	Research Topic	Research Methodology	Result
Harb, et al., 2015	Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case.	The analysis presented a multi-criteria decision approach	The analysis presented a multi-criteria decision approach to systematically frame the methodology-selection problem teaching case designed to help Information Systems students improve their skills in understanding and evaluating complex business requirements and in selecting the most appropriate software development methodology to match the needs of a specific IT project, and the organization. The teaching case includes a comparative overview of various agile methodologies, as well as the use of multi-criteria decision tools for solving the problem of methodology selection. a multi-criteria decision approach to systematically frame the methodology-selection problem
Kamal, 2015.	Developing a sustainability network for information technology adoption and use in micro-enterprises.	An action research methodology approach was employed	In investigating the extent to which Information Technology adoption and usage could be sustained in micro-enterprises so as to facilitate business growth. The study developed an online tool to facilitate micro-enterprises' sustainability of ICT adoption and use. However, this study does not provide a

Author	Research Topic	Research Methodology	Result
			framework for the adoption of software development methodologies.
(Moe *et al.*, 2012)	Challenges of shared decision-making: A multiple case study of agile software development.	The study applied qualitative approach	Results identified three major challenges associated with shared decision-making in agile software development, i) alignment strategic product plans to iteration plans, ii) development resources allocation, iii) performing development and maintenance tasks in teams. The limitation of this study is that it only focused on software product companies using Scrum.
Sheffield & Lemétayer 2013	Factors associated with the software development agility of successful projects	The study was anchored on qualitative approach, employing interviews methods, which used the card sort technique.	Analysis of the survey data showed that software development agility was directed by a project environment factor (organizational culture) and a project factor (empowerment of the project team). While these results aimed to assist practitioners to reflect on development practices, and negotiate change so as to achieve higher rates of project success. The study was Eurocentric based as it was conducted in Wellington, New Zealand.
Vijayasarathy & Butler,	Choice of software development	The design of this empirical study is quantitative in nature.	Study findings revealed that while agile methodologies such as Agile Unified Process and

Author	Research Topic	Research Methodology	Result
2016).	methodologies: Do organizational, project, and team characteristics matter?	Through an anonymous online survey data was collected	Scrum have been dominant over 10 years ago, traditional methodologies, like waterfall model, are still popular and are in use today. Likewise, organizations are taking a hybrid approach, using multiple methodologies on projects. Besides, their choice of methodologies is associated with certain organizational, project, and team characteristics.
Passos *et al.* (2014)	The role of organizational culture in software development practices: a cross-case analysis of four software companies	The study employed qualitative thematic analysis	Study findings revealed strong influence of past experiences and organizational contexts on software development practices. This study was conducted in Brazil and not in South Africa.
Khoza and Pretorius (2017)	Factors negatively influencing knowledge sharing in software development	The study was anchored on a quantitative approach and data were collected using an online questionnaire with closed-ended questions	Findings showed that job security, motivation, time constraints, physiological factors, communication, resistance to change and rewards are vital factors negatively influencing knowledge sharing in software organizations.
(Roses *et al.*, 2016)	Favorability conditions in the adoption of agile	The methodology used in this study is a mixed method based	Based on the findings a model was proposed to assess the degree of favorability conditions

Author	Research Topic	Research Methodology	Result
	method practices for software development in a public banking	approach. This research was conducted in cooperation with software developers of a Brazilian public retail bank	in the adoption of Agile practices. This study limitation was its focus on only traditional software methodologies.
(Rajagopalan & Mathew, 2016)	Choice of agile methodologies in software development: A vendor perspective	The methodology employed a desktop approach	Findings on Framework 1 showed that the choice of XP for one project was not supported base of framework guidelines. While the choices of SCRUM for other two projects, were partially supported. The study shortcoming was its focus on only agile methodologies
(Lee *et al.*, 2016)	Examining the impacts of organizational culture and top management support of knowledge sharing on the success of software process improvement	The study used partial least squares (PLS) statistics to analyze 118 samples from SPI-certified Taiwanese organizations so as to develop an innovative model for explore the success of software process improvement (SPI	Findings suggest that that clan-type organizational culture has a stronger relationships with knowledge sharing than hierarchy-type in the context of SPI success. SPI knowledge sharing is indicated to be a mediator of between clan culture and top management support within the context of SPI success.
(Lalsing *et al.*, 2012)	People factors in agile software development and project management.	The methodology of the study was anchored on survey using a quantitative approach	The results of the study clearly show that for agile methodologies to work well, it is important to select the right people for the right team.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 INTRODUCTION

The methodology and design employed in this study aligns the research objectives, research questions, data collection methods and data analysis. The aim of the chapter is to describe data collection methods, offer justification for the selected research design, research instruments, and data collection techniques used in the study. Likewise, associated issues encountered during data collection process are explained. The chapter considers the philosophical underpinning of the study, research approach and design, research methodology and empirical research. Reliability and validity and ethical considerations were further discussed. The next section present the contents of this chapter in the form of a document map.

3.2 RESEARCH METHODOLOGY CONTENTS MAP

A document map describes the relationships or topics under construction that inform the chapter topic (Chukwuere, 2016 & Emekako, 2015).While the document map is often represented in a diagrammatic format, Novak and Cañas (2008) claim that the document map al the blueprint of a chapter to be seen, understood as well as to showcase concept and structure. This is to say that, the document map is a structural graphical representation of the flow of ideas covered in a document. Likewise, it is a directive that connects the flow of contents within a document together, by linking one area of knowledge to another in a logical and scientific manner.

Figure 10 below is the chapter document map of the research methodology employed by the researcher. It indicates the flow straight from the philosophical underpinnings of the study to the benchmark of thesis evaluation.

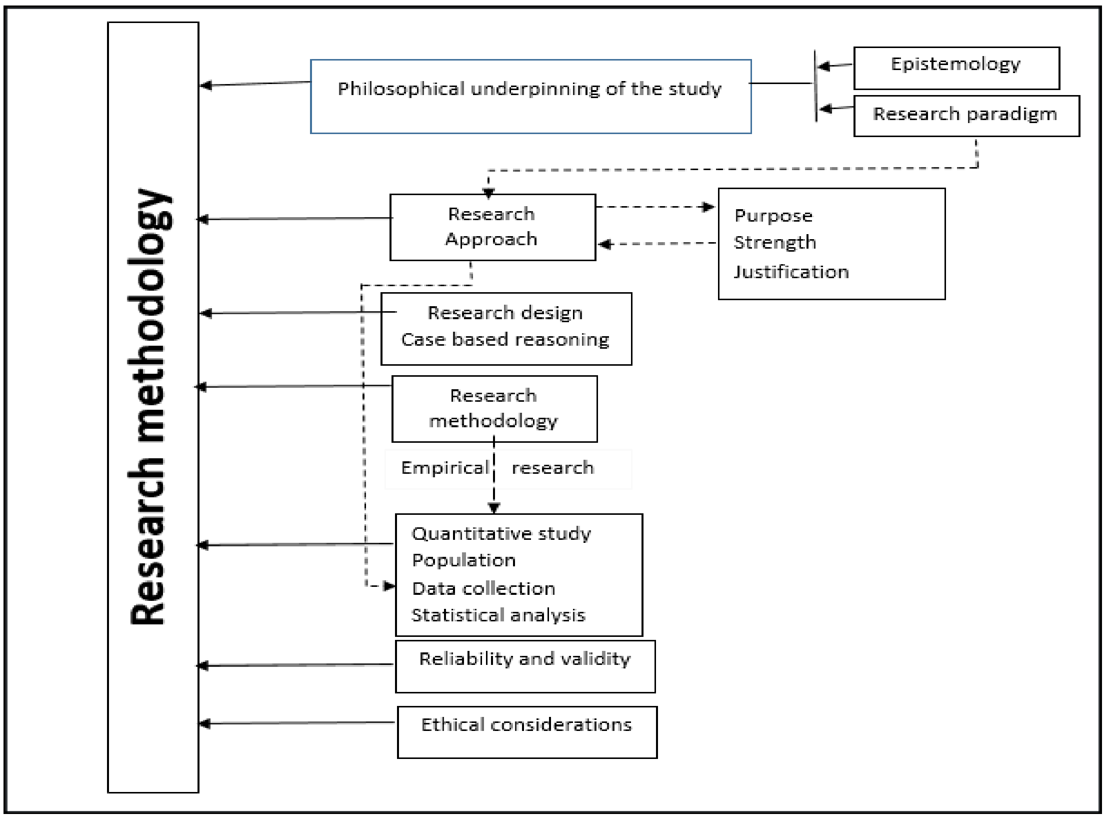


Figure 10: Research Methodology Content Map (Babbie E & Mouton J, 2001)

3.3 PHILOSOPHICAL FOUNDATION OF THE STUDY

This study aims at developing a framework to assist organizations in making a sound decision regarding which software development methodology to apply during their software development lifecycle. To achieve this, a systematic explanation of the philosophical worldview of epistemology and paradigm is necessary.

3.3.1 Epistemology

Past scholars have described epistemology to be the theory of knowing (Marsh & Furlong, 2002). Epistemology assists to determine how people really know or confirm what is true, particularly how this truth could be validated according to different disciplines (Sturgeon, Martin & Grayling, 1995). Different discipline may hold a different reality which employs different epistemologies as the research domains may differ (Elliot, 2002). This present study focuses on epistemology that investigates or validates truths in the sciences relating to information science.

Every discipline arrives on norms concerning how researchers should collect data, validate and present data and even judge theories. According to scientist philosopher, Jane Maienschein:

"It is epistemic convictions that dictate what will count as an acceptable practice and how theory and practice should work together to yield legitimate scientific knowledge" (Maienschein, 2000:123).

In like manner, Repko (2012) states that over the years, an investigator's epistemological stance is shown in what is studied and how it is studied. These epistemological viewpoints includes: positivist, constructivist/interpretivist, transformativist and the postcolonial worldviews (Creswell, 2014). See Table 8 below for a comparison of selected paradigms. This study focuses on the positivist worldview. The study uses epistemological positivism which assumes that the only way to establish truth and objective reality is through the scientific method. Table 8 below shows selected epistemology with justifications of choosing any particular one, alongside their philosophical underpinnings, ontological assumptions for adopting any particular worldview and most applicable methodology.

Table 8: Comparison of Selected Paradigms

	Positivist/ post positivist paradigm	**Constructivist/ interpretative paradigm**	**Transformative / emancipatory paradigm**	**Postcolonial/ indigenous research paradigm**
Reason for doing the research.	To discover laws that are generalizable and govern the universe.	To understand and describe human nature.	To destroy myths and empower people to change society radically.	To challenge deficit thinking and pathological descriptions of the former colonized and reconstruct a body of knowledge that carries hope and promotes transformation and social change among the historically oppressed.
Philosophical underpinnings	Informed mainly by realism, idealism and critical realism.	Informed by hermeneutics and phenomenology.	Informed by critical theory, postcolonial discourses, feminist theories, race specific theories	Informed by indigenous knowledge systems, critical theory, postcolonial discourses, feminist

	Positivist/ post positivist paradigm	**Constructivist/ interpretative paradigm**	**Transformative / emancipatory paradigm**	**Postcolonial/ indigenous research paradigm**
			and neo-Marxist theories.	theories, critical race- specific theories and neo-Marxist theories.
Ontological assumptions.	One reality, knowable within probability.	Multiple socially constructed realties.	Multiple realties shaped by social, political, cultural, economic, race, ethnic, gender and disability values.	Socially constructed multiple realities shaped by the set of multiple connections that human beings have with the environment, the cosmos, the living and the non-living.
Place of values in the research process.	Science is value free, and values have no place except when choosing a topic.	Values are an integral part of social life; no group's values are wrong, only different.	All science must begin with a value position; some positions are right, some are wrong.	All research must be guided by a relational accountability that promotes respectful representation, reciprocity and rights of the researched.
Nature of knowledge.	Objective.	Subjective; idiographic.	Dialectical understanding aimed at critical praxis.	Knowledge is relational and is all the indigenous knowledge systems built on relations.
What counts as truth?	Based on precise observation and measurement that is verifiable.	Truth is context dependent.	It is informed by a theory that unveils illusions.	It is informed by the set of multiple relations that one has with the universe.
Methodology.	Quantitative; correlational; quasi-experimental; experimental; causal	Qualitative; phenomenology; ethnographic; symbolic interaction; naturalistic.	Combination of quantitative and qualitative action research; participatory research.	Participatory, liberating, and transformative research approaches and methodologies

	Positivist/ post positivist paradigm	**Constructivist/ interpretative paradigm**	**Transformative / emancipatory paradigm**	**Postcolonial/ indigenous research paradigm**
	comparative; survey.			that draw from indigenous knowledge systems.
Techniques of gathering data.	Mainly questionnaires, observations, tests and experiments.	Mainly interviews, participant observation, pictures, photographs, diaries and documents.	A combination of techniques in the other two paradigms.	Techniques based on philosophic sagacity, ethno philosophy, language frameworks, indigenous knowledge systems and talk stories and talk circles.

3.3.2. Research paradigm

The researchers choose to apply positivism worldview as a paradigm for this study because this paradigm promotes the use of quantitative approach as showed in Table 8 above. More so, positivism otherwise known as logical positivism proposes that the only way to establish the truth is through scientific approach, the study is an attempt to establish the truth with regard to choosing the right SDM to apply during software development lifecycle. Likewise, another justification for applying positivism is because the paradigm is anchored on the opinion that science is the only foundation for true knowledge. It further concludes that, the methods, techniques and procedures applied in the natural sciences is the most suitable framework for researchers to investigate the social world.

According to Crotty (1998) positivism is a reflection of a strict empirical approach whereby claims about knowledge are strictly based on experience. Additionally, it stresses facts and the causes of behaviour. Likewise positivism naturally relates the scientific method to the study of human action. Positivism today is seen as being objectivist, in order words, objects around people have existence and meaning, and they are independent of their consciousness of them (Crotty, 1998).

3.4 RESEARCH APPROACH

The idea behind the choice of quantitative approach is that it is an inquiry into a social or human problem, based on testing a theory composed of variables, measured with numbers, and analysed with statistical procedures, in order to determine whether predictive generalisations of the theory hold true (Creswell, 2007). In addition, in the positivism paradigm, a research purpose is to predict results, test a theory, and find the strength of associations between factors, or conduct a cause and effect relationship. In this regard, quantitative approach starts with ideas, theories or concepts which are defined as they are used in the study to point to the variables of interest.

3.4.1 Proposed Framework

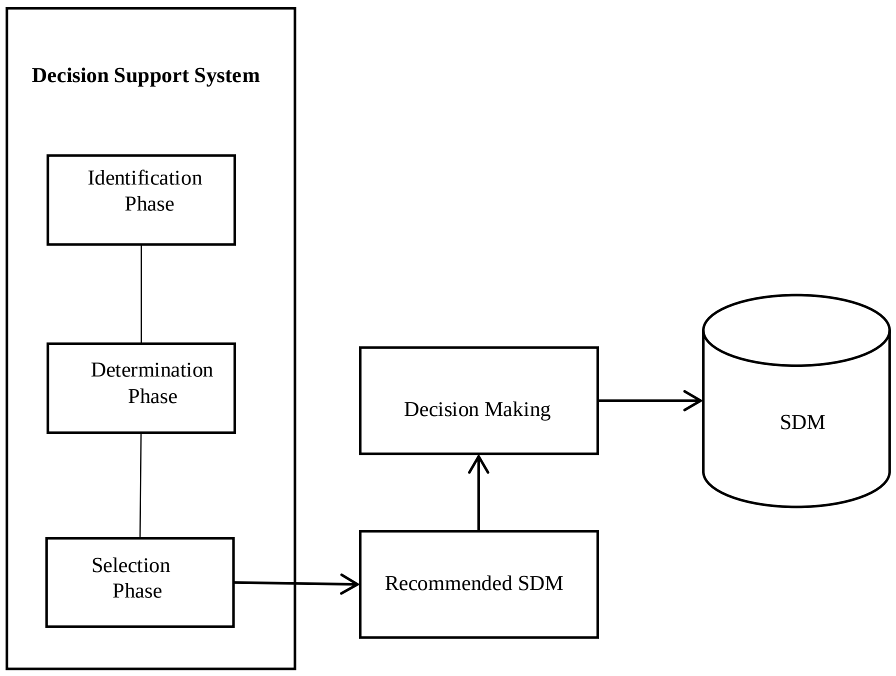


Figure 11: Proposed Framework (Verma, Bansal and Panley, 2014)

Figure 11 illustrates the proposed framework for this study; the phases of the framework are detailed below.

Identification Phase: Identification is the requirements analysis step carried out in traditional software development. It involves a formal task analysis to determine the external requirements, the form of the input and output, the setting where the program will be used; and also determines the users of the system (Sylvester 2014).

Determination Phase: This involves organizing the key concepts, sub-problems and information flow into formal representations. In effect, the program logic is designed at this stage (Sylvester 2014).

Selection Phase: No one process is ideal so a framework is developed for picking a process which depends on multiple components, project characteristic and selection boundaries. These selection boundaries are: requirement specification, complexity of system, time, size and change incorporated/user impact (Choudhary, Kasgar, and Kashyap 2015).

3.5 RESEARCH DESIGN

The study was anchored on a Case Based Reasoning (CBR) methodology because it focuses on problem solving that centres on the reutilization of past experience (Aamodt & Plaza, 1994). It is based on solutions, information and knowledge available in similar problems previously solved. The CBR methodology was implemented using the SQL programming language. A CBR production rule uses First Order Logic for knowledge representation and is structured in two parts:

IF

<<Conditions>

Then

<<Actions>>

The assumption of CBR is that, remembering, understanding, experiencing, and learning is not separated from each other. More so, human memory is dynamic which often change because of experiences. Therefore, the justification for using CBR in the present study is that it improves people's performance as to become more efficient by remembering old solutions given to similar problems and adapting these solutions to fit a new problem instead of solving it from scratch. This is particularly relevant as different methodologies for software development have been introduced over the last three decades (Abrahamsson *et al.*, 2017b; Despa, 2014; Tavares *et al.*, 2017).

These different methodologies requires constant adaptation to information systems so as to reflect changes in the type of information needed as a result of changes in technology, business processes, structure, or the external environment (Zykov, 2018). CBR becomes important because its elements of remembrance, understanding and experience are useful for learning the latest Software methodologies. According to Steels (1990) CBR augments the ideas about the components of expertise using the solved cases as an episodic memory. Likewise, case-based reasoners become more competent in their evolution over time, this allows for deriving better solutions especially when faced with less experienced situations, the implication of this is that it assist in preventing people from repeating the same mistakes in the future (learning process).

All case-based reasoning methods have in common the following process (Shekapure & Nagar, 2015):

- Retrieve the most similar cases as stored in the case library;
- Reuse the retrieved case in an attempt to solve the present problem;
- Revise and adapt the proposed solution to fit current situation; and
- Retain the final solution as part of a new case and reuse in other similar cases

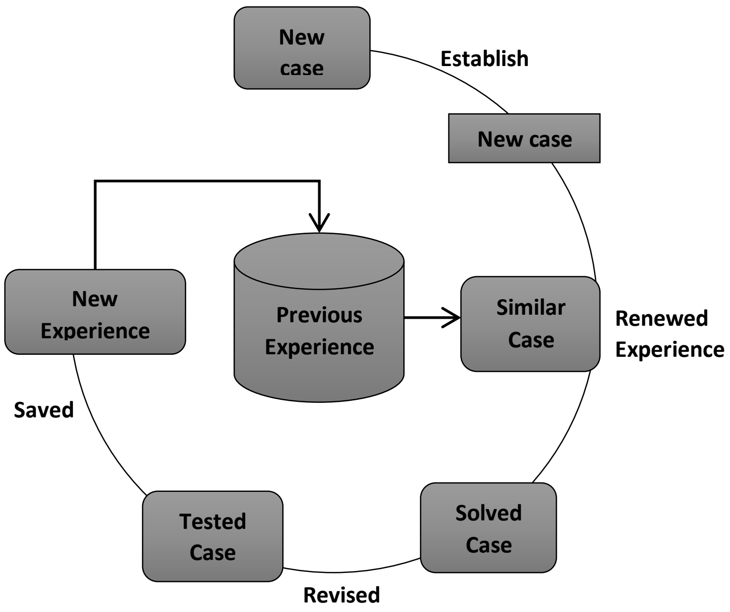


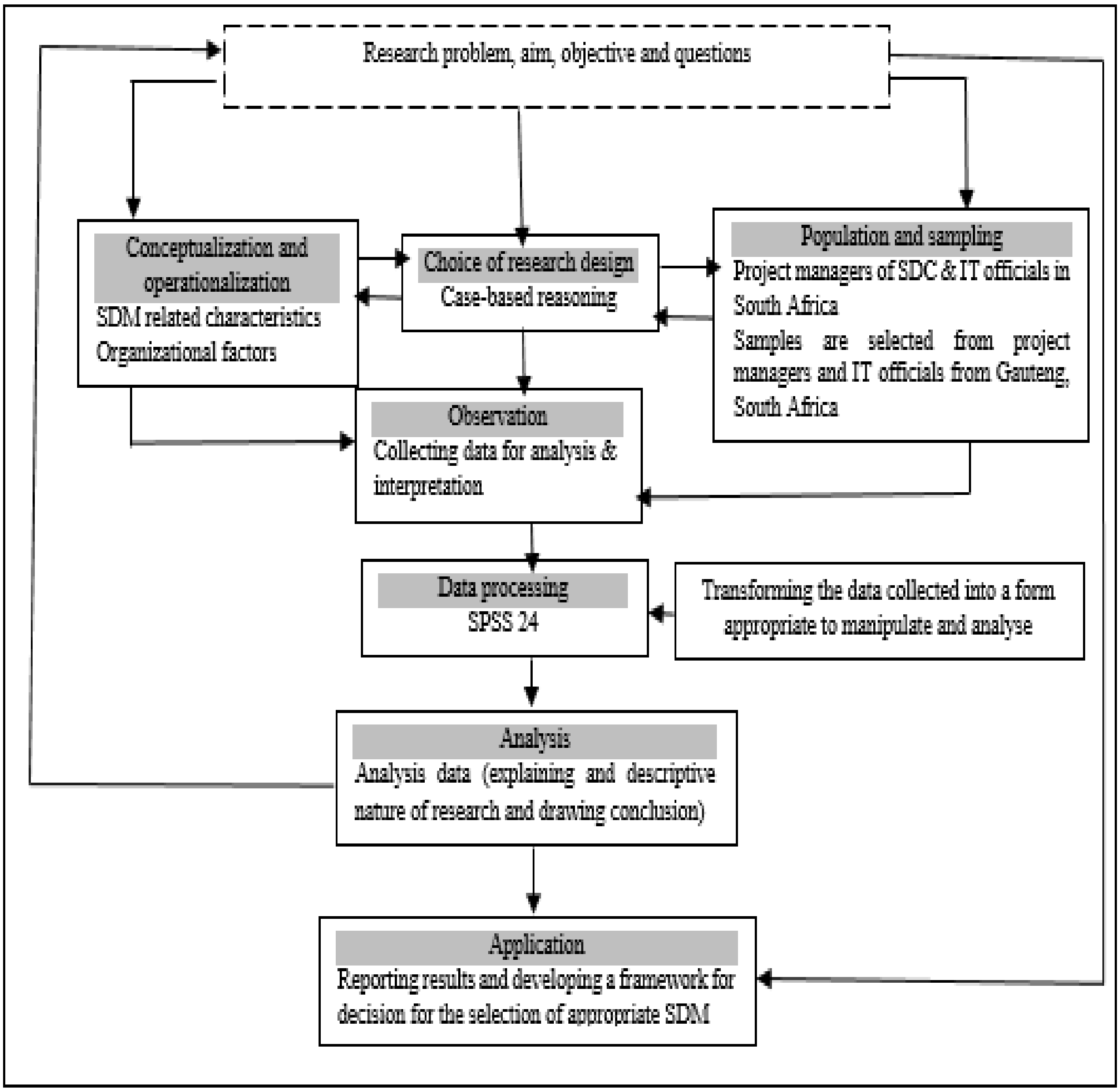
Figure 12: Case Based Reasoning (Shekapure & Nagar, 2015)

As depicted in Figure 12 above, case-based reasoning is a problem solving mechanism that is fundamentally different from other major problem solving methodologies. Instead of relying solely on general knowledge of a problem, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of past experiences and concrete problem cases.

A new problem is solved by finding similar problems from the past, and reusing it in the new problem context. A second important difference is that CBR is also an approach to incremental and sustained learning. This is because a new experience is retained each time a problem has been solved, making it immediately available for future cases.

3.6 METHODOLOGICAL PROCESS

Figure 13 below describes the schematic methodological process of the empirical process for the study. This figure was adapted from Babbie, E. & Mouton, J. (2001) and was modified to fit the focus of the present study.



Research problem, aim, objective and questions
Conceptualization and operationalization
SDM related characteristics
Organizational factors
Choice of research design
Case-based reasoning
Population and sampling
Project managers of SDC & IT officials in South Africa
Samples are selected from project managers and IT officials from Gauteng, South Africa
Observation
Collecting data for analysis & interpretation
Data processing
SPSS 24
Transforming the data collected into a form appropriate to manipulate and analyse
Analysis
Analysis data (explaining and descriptive nature of research and drawing conclusion)
Application
Reporting results and developing a framework for decision for the selection of appropriate SDM

Figure 13: Showing methodological Process of Study (Babbie, E. & Mouton, J., 2001)

3.7 EMPIRICAL RESEARCH

The empirical research of this quantitative study entails a study of the population, sampling techniques and statistical analysis.

3.7.1 Population

The population is the targeted community from which the sample for a study is selected (McMillian & Schumacher, 2010). The entire population is not always studied but only part of it. Population of a study habitually possesses same or similar characteristics and the researcher task is to choose from this population the characteristics relevant and useful for the study through means of scientific sampling techniques. Because this study investigates a decision support framework for software development methodologies for IT related organizations in South Africa, the characteristics essential for choosing participants in this study were variables related to software development methodologies and working environment in South Africa, particularly in Gauteng Province. The population of this study is all software development firms in Gauteng, South Africa. This includes both listed and unlisted companies. This however; is only limited to companies that operate in South Africa.

3.7.2 Sampling Technique

O'Leary (2010:162) defined sampling as a process of selecting elements of a population so as to participate in a research study, since it is almost impossible to use an entire population, a sample was drawn from the study population described in section 3.7.1 in order to infer and arrive at a conclusion.

Convenience sampling and purposive sampling techniques were employed as sampling technique in this study. Convenience sampling was used as a sampling technique by the researcher to select ten software development companies based in Gauteng province, using their proximity to the major roads as a criterion. Easy accessibility and cost were crucial factors to be considered. According to Maree (2010:177), convenience sampling is described as a sampling methods based on availability, accessibility and cost.
Purposive sampling, otherwise called judgmental sampling (Babbie, 2007:184) was also used for selecting participants for the study. Thus, project managers and staff working in software development companies were purposively selected to participate in this study. This sampling technique is when researchers choose participants because they are most appropriate for

providing the best answers to the subject under investigation (Babbie, 2007:184). The justification for using purposive sampling for this study is that the method considers the selection of participants, the settings, incidents, events and activities for data collection (Maree, 2010:178), all which are relevant for the researcher.

A total of 100 participants were selected to participate in this quantitative study. These participants are made up of professionals within the software development industry in Gauteng province. These include project Managers, business analysts, developers and software testers.

Each participant will report on a software development project which they have worked on. Therefore; 100 projects will be employed as training data for this study. The data from these 100 projects will be used to formulate the decision support framework for the adoption of a software development methodology.

Furthermore; 15% of this data will be subsequently used to test the accuracy of the decision support framework (DSS). .The sampling procedure used is shown below.

3.8 DATA COLLECTION PROCEDURE

The case based reasoning methodology was used in combination with a Likert Scale; this is a type of rating scale which is used to measure the performance directly. Likert Scale is a five point rating scale which allows the user to express their thoughts based on how much they agree or disagree with a problem statement. Users give their responses by choosing a Likert item. A Likert item is a word or statement which the user is asked to evaluate according to the given criteria. Likert items are used to measure the degree of agreement or disagreement. In this study we used a Likert scale questionnaire that was adopted from Verma, Bansal, and Pandey (2014). See sample questionnaire in Table 10 below.

Table 7: Likert Scale Questionnaire (Verma, Bansal, and Pandey 2014)

Features	Poor	Fair	Average	Good	Excellent
Required Specification	1	2	3	4	5
Complexity of system	1	2	3	4	5
Time Schedule	1	2	3	4	5
Cost	1	2	3	4	5
Documentation	1	2	3	4	5
Project Size	1	2	3	4	5

Features	Poor	Fair	Average	Good	Excellent
Change Incorporated	1	2	3	4	5

The questionnaire was distributed to 100 participants in ten software development companies within Gauteng Province. Before the questionnaire was given to participants, the researcher explained the study purpose in detail. Upon creating rapport with the researcher, and establishing willingness to participate in the study, participants signed the consent form and all ethical principles were explained to them. Participants were informed that participation was voluntary and they have the right to decline and even stop mid-way into the study without being penalized. Data collection process took six weeks and 7-10 minutes was used to complete each questionnaire.

The proposed framework will be rule based, and will be generated with the assistance of a Likert scale measurement.

Example of a rule based model

IF Requirement specification <= 1 and
Complexity of system <= 2 and
Time schedule <= 5 and
Cost <= 2 and
Documentation <= 5 and
Project size <= 4 and
Change incorporated <= 1 and
THEN Waterfall Model (Score <= 20)

Data collected were coded and analyzed on SPSS version 24. SPSS software is statistical software for analyzing quantitative data. Question one and two were tested using descriptive analysis. Additionally, univariate analysis was conducted on the raw data and tables with graphs were presented. Based on the findings of the analysis, proposed framework was developed to determine the most appropriate software methodology to be adopted for the projects.

3.9 EVALUATION OF THE FRAMEWORK

There are four key issues that may influence the quality of this research.

3.9.1 Research Validity

Validity is essential in research because it relates to whether you are actually measuring what you planned to measure (Pearson, 2010). It is concerned with the accuracy of the findings. The questionnaire used in this study was adopted from research done by Verma, Bansal, and Pandey (2014), therefore the questionnaire is deemed valid.

In this study, the researcher ensured validity of the results by employing the **Scenario Based Simulation** technique. This tests the proposed framework in action and demonstrates the effectiveness of the proposed framework in a real world environment. The researcher will also ensure validity of the results by comparing results to existing evidence by other researchers.

3.9.2 Research Reliability

Reliability relates to whether, if you carried out the research again, you would get the same or similar results (Pearson, 2010) . In this study, the researcher will ensure reliability of the results by making use of multiple repetitions of measurement over a long period of time, at different points of time, in different scenarios or settings and by different persons.

3.9.3 Research Accuracy

Accuracy relates to how close your measurement is to the 'gold standard', or the intended result (Pearson, 2010). In this study, the researcher will ensure accuracy of the results by testing the precision of the Decision Support Framework.

3.9.4 Research Precision

Precision is related to the refinement of the measuring process. It is concerned with how small a difference the measuring device can detect (Pearson, 2010). In this study, the researcher will ensure precision of the results by making use of a **Rules Based Reasoning** approach that focuses on the reutilization of past experiences. In this study we will test the accuracy of the framework by applying rules based reasoning to real life software development projects that have already been implemented.

3.10 ETHICAL CONSIDERATIONS

Ethical issues are present in any kind of research. The research process creates tension between the aims of research to make generalizations for the good of others, and the rights of participants to maintain privacy. Ethics pertains to doing good and avoiding harm (Orb et al., 2000).

With the above understanding the following ethical considerations were well-thought-out during this study. Permission to conduct the study was sought and obtained from the ethics committee of the Vaal University of Technology. Likewise, the researcher obtained permission to conduct the study from stakeholders within the software development companies, this was important for a smooth data collection process.

In addition, participation in the study was voluntary, therefore, participants who indicated willingness to participate in the study read and signed the Informed Consent Form. The form specified the purpose of the study, and described the core fundamental principles of ethics. In this regards, the researcher is aware of the responsibility of conducting research in accordance with the ethical principles as endorsed by the Vaal University of Technology. Ethical considerations are the fundamental principles upon which human subject protections are based and they are as follows:

1. **Respect for Persons**: Ethical research honours the autonomy of individuals to make an informed choice about participation in research and provides suitable protection for vulnerable persons. Thus, participation was not only voluntary but participants had the right to withdraw their participation from the research whenever they felt uncomfortable with any aspect of it.

2. **Beneficence:** Ethical research has scientific or scholarly value in which the potential benefits outweigh the risks, which are justified and minimized. In this regards, the principle of beneficence which covered the aspect that this study had significant benefit to both participants and the society at large was noted.

3. **Justice**: Ethical research is designed and conducted so that the burdens and benefits are fairly distributed regardless of age, race, gender, ethnicity, etc. The principle of justice which had great consideration to respect all participants equally was adhered to. In light of this, study participants were assured of confidentiality of their responses. Names and identification numbers were not required.

3.11 SUMMARY OF THE CHAPTER

The chapter has presented a detailed comprehensive research methodology of how data was collected for the study. The research methodology approach guides the research process and the kinds of tools and procedures used in order to achieve the aim of the study. This process involves research approach, design and methods, data gathering methods and statistical analysis. Likewise, this data was not only collected but analysed. The next chapter will present the main results of this research detailing all statistical tests used in analysing the stated questions.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 INTRODUCTION

The purpose of this study is to formulate a decision support framework for the adoption of software development methodologies in South Africa. This chapter describes the analysis of data followed by a discussion of the research findings. In this regard, this chapter reports on analysis of data which was obtained from questionnaires completed by 90 professionals; a 90% response rate was achieved in this study. The aim of the study is to determine the best methodology to adopt in a software development project. It should be emphasized that the findings presented in this chapter provide answers to the main research question. The Findings is discussed by linking them to existing literature. The chapter starts with presentation of results, followed by its interpretation and discussion.

4.2 PRESENTATIONS OF RESULTS

The graphs below (Figures 14 to 19) are a presentation of the data collected from the questionnaires. This data depicts the average scores that were received for each project characteristic in relation to a software development methodology that was applied in a project. This data was subsequently used to formulate the decision support framework rules/algorithm.

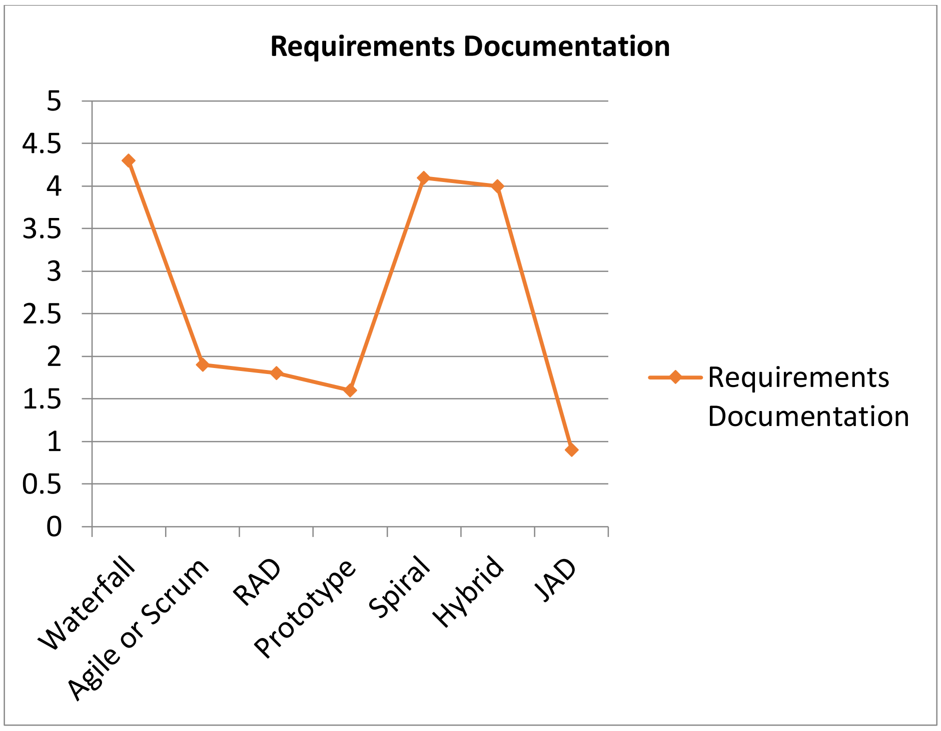


Figure 14: Average score for requirements documentation

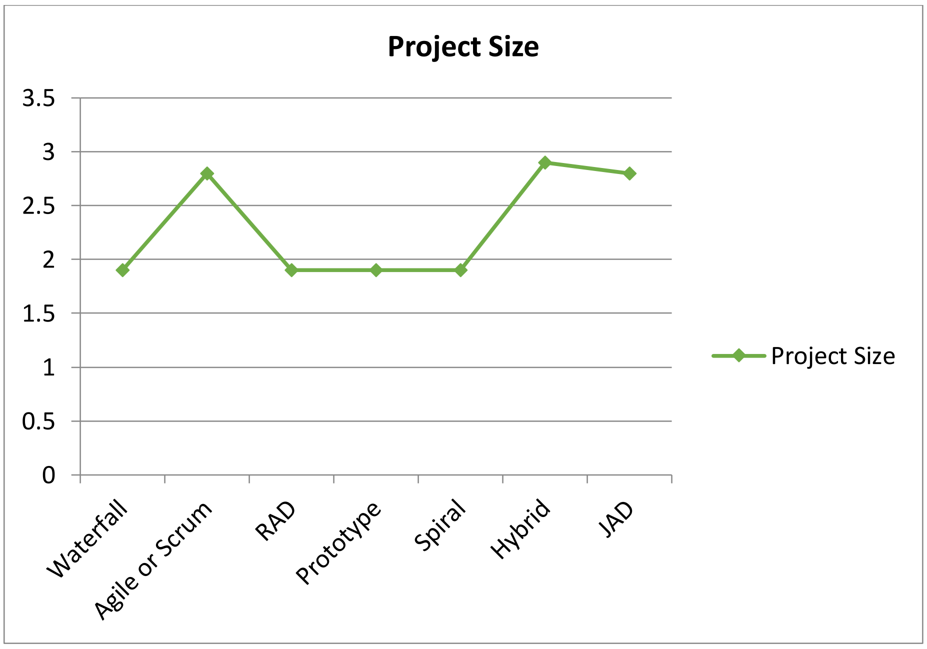


Figure 15: Average score for project size

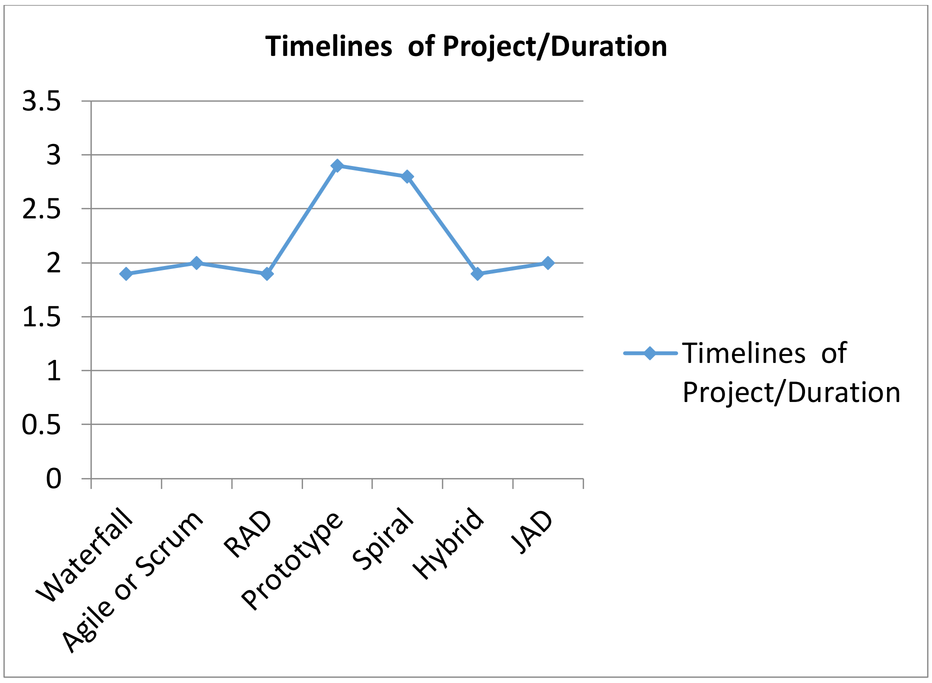


Figure 16: Average score for project timelines

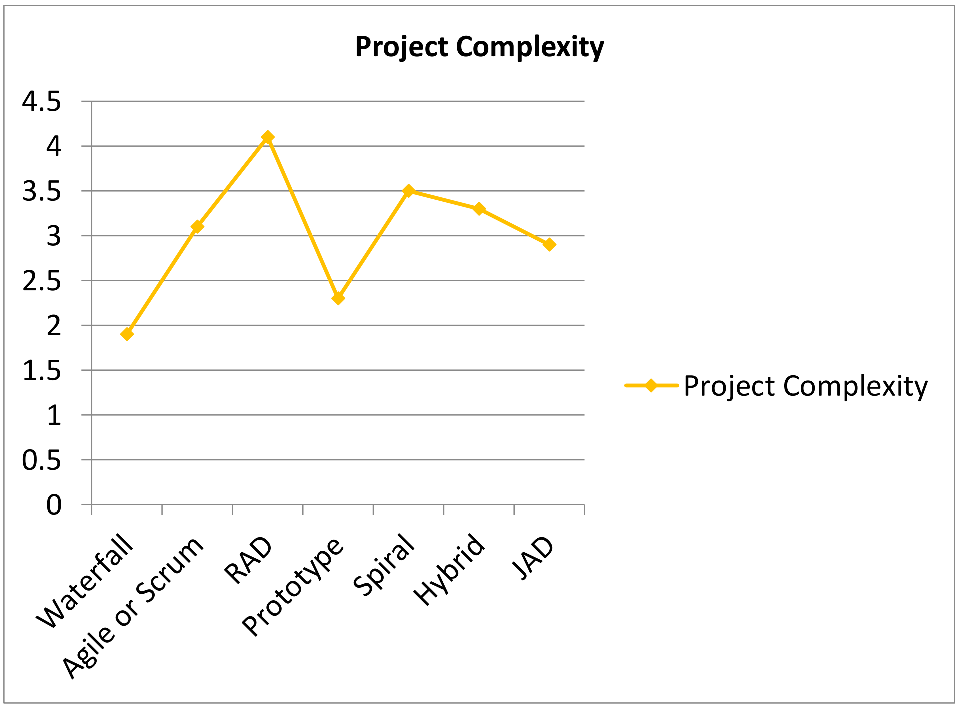


Figure 17: Average score for project complexity

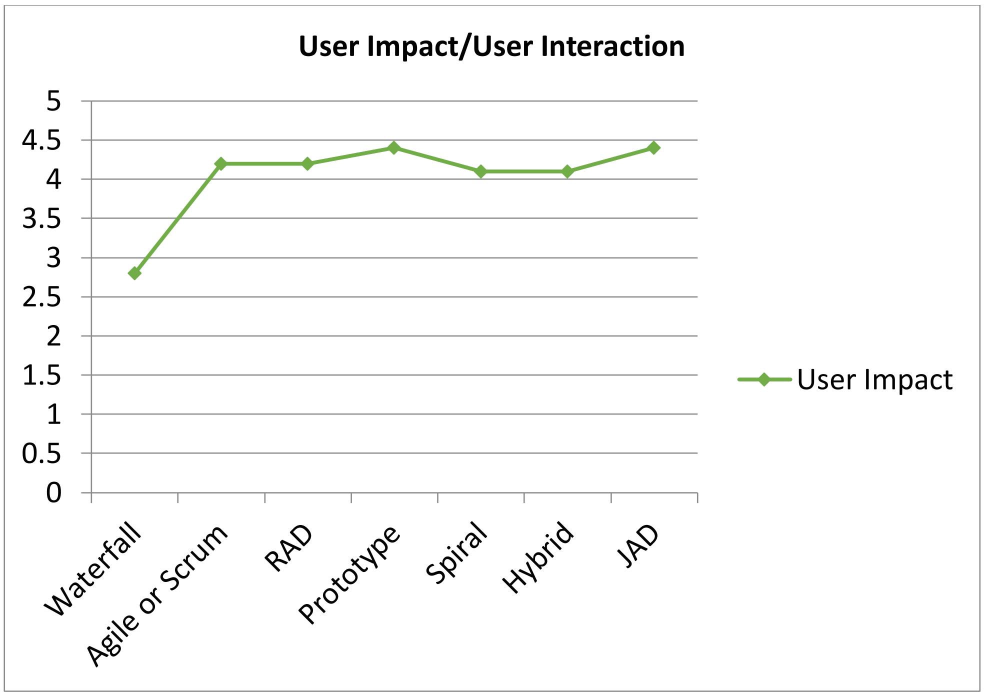
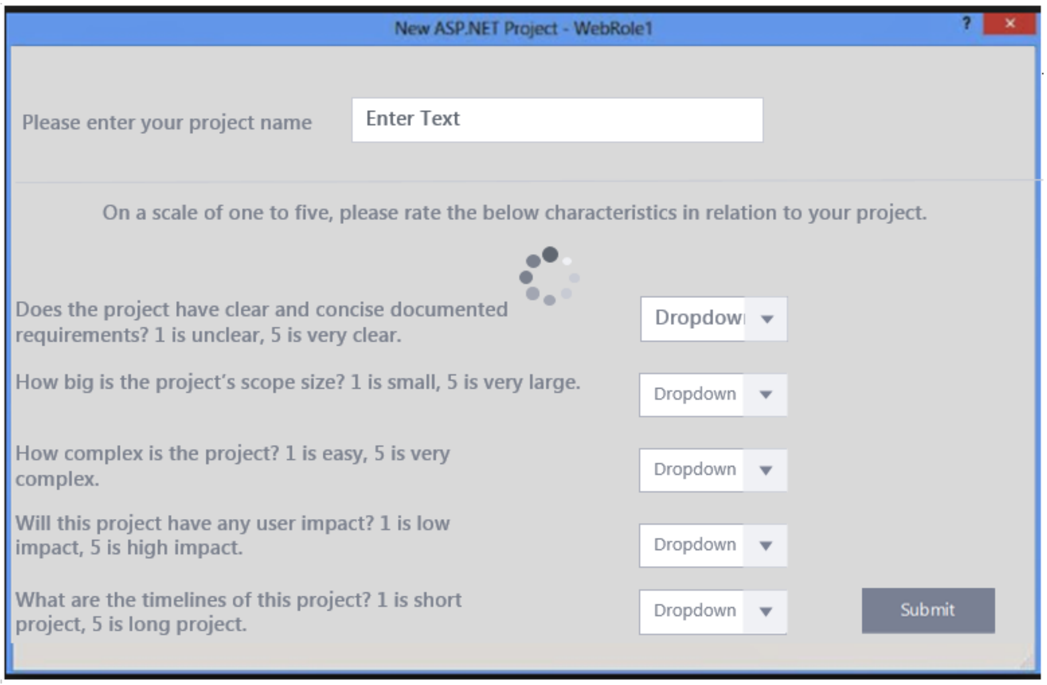


Figure 18: Average score for user impact

Figure 19 below is a presentation of the Decision Support Framework user interface. This framework is made up of a user interface as well as back end rules. Users will be required to answer questions presented by the DS Framework user interface. Upon submission, the DSS will then present the best software development methodology to adopt for the specific project. The DSS is built on a SQL rules foundation. These rules were formulated using data collected from the questionnaires.



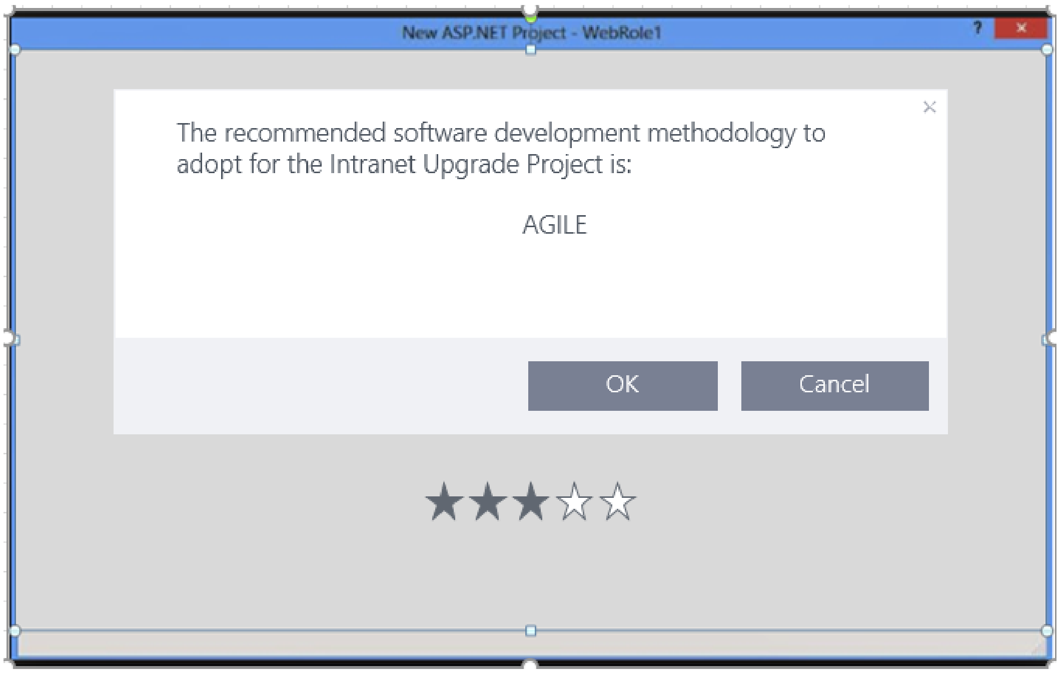
Please enter your project name

Enter Text

On a scale of one to five, please rate the below characteristics in relation to your project.

Does the project have clear and concise documented requirements? 1 is unclear, 5 is very clear.	Dropdown
How big is the project's scope size? 1 is small, 5 is very large.	Dropdown
How complex is the project? 1 is easy, 5 is very complex.	Dropdown
Will this project have any user impact? 1 is low impact, 5 is high impact.	Dropdown
What are the timelines of this project? 1 is short project, 5 is long project.	Dropdown

Figure 19: Decision Support Framework user interface



The recommended software development methodology to adopt for the Intranet Upgrade Project is:

AGILE

★★★☆☆

Figure 20: Decision Support Framework user interface - continued

Below is a presentation to the DSS rules. These rules were formulated based on data collected from the questionnaire. The rules are the engine of the Decision support framework.

```
SET SERVEROUTPUT ON

DECLARE

   v_Req NUMBER ;

   v_size NUMBER;

   v_complexity NUMBER

   v_userimpact NUMBER

   v_timelines NUMBER

BEGIN

  IF v_Req >=4

       AND v_size <=2

       And v_complexity <=2

       AND v_userimpact <=3

       AND v_timlelines <=2

 THEN

      DBMS_OUTPUT.PUT_LINE (‘WATERFALL’);

 ELSE

      IF v_Req <=2

          AND v_size <=3

         AND v_complexity >=3

         AND v_userimpact >=4
         AND v_timlelines <=2

THEN

      DBMS_OUTPUT.PUT_LINE (‘AGILE’ or ‘SCRUM’);

ELSE

          IF v_Req <=2

            AND v_size >1

            AND v_complexity >=4

            AND v_userimpact >=4

            AND v_timlelines <=2

THEN

      DBMS_OUTPUT.PUT_LINE (‘RAD’);

ELSE

                IF v_Req <=2

                  AND v_size >1

                  AND v_complexity >2

                  AND v_userimpact >=4

                 AND v_timlelines <=3

THEN

      DBMS_OUTPUT.PUT_LINE (‘PROTOTYPE’);

ELSE
```

```
                IF v_Req >=4
                  AND v_size >1
                  AND v_complexity >3
                  AND v_userimpact >=4
                  AND v_timlelines <=3
THEN
    DBMS_OUTPUT.PUT_LINE (‘SPIRAL’);


ELSE
                  IF v_Req >=4
                    AND v_size <=3
                    AND v_complexity >=3
                    AND v_userimpact >=4
                    AND v_timlelines <=2
THEN
    DBMS_OUTPUT.PUT_LINE (‘HYBRID’);
ELSE
                      IF v_Req <=1
                        AND v_size <=3
                        AND v_complexity >=3
```

```
                         AND v_userimpact >=4

                         AND v_timlelines >=2




THEN

   DBMS_OUTPUT.PUT_LINE (‘JAD’);



                           END IF;

                         END IF;

                     END IF;

                  END IF;

              END IF;

     END IF;

END IF;

END;
```

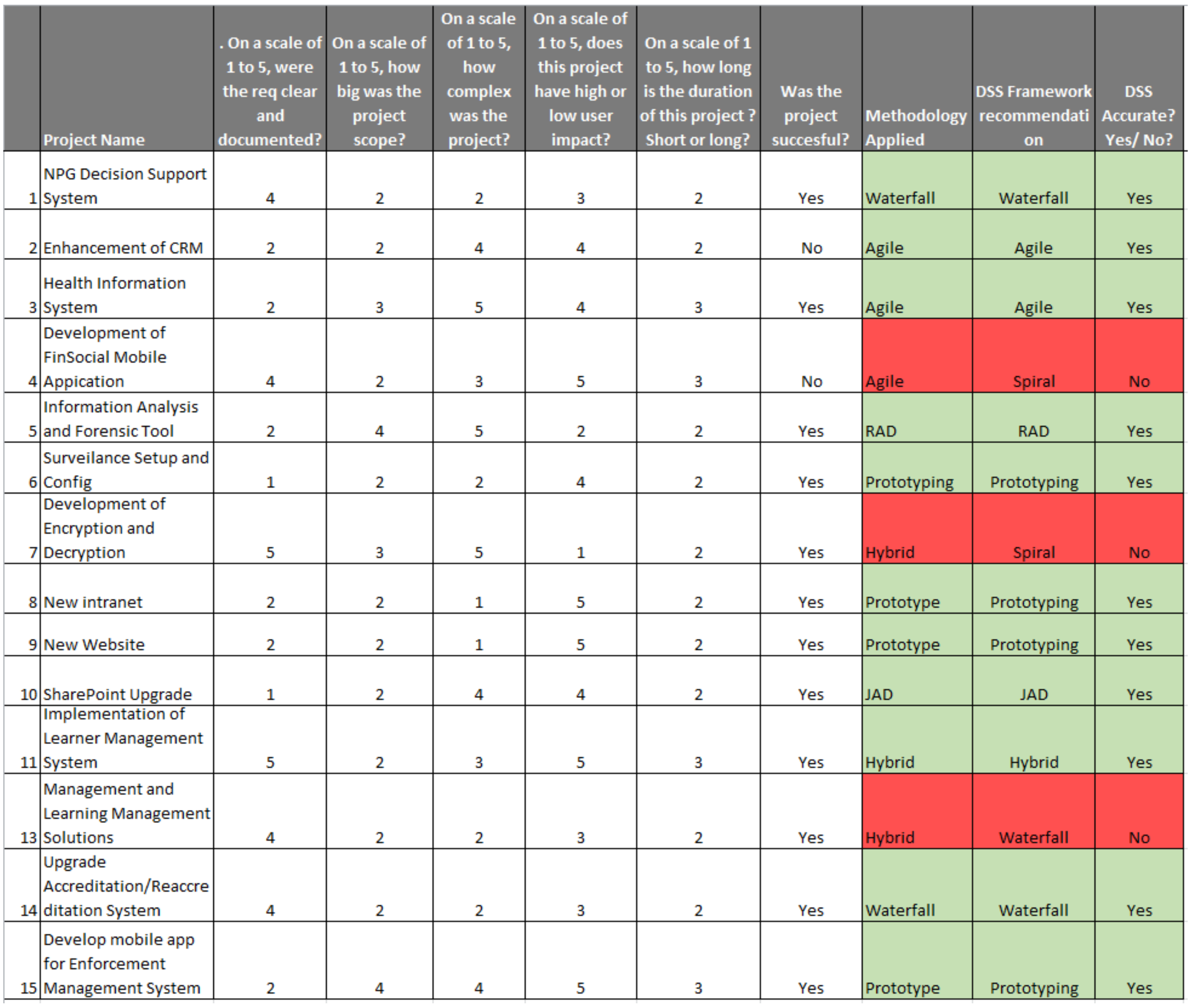
4.3 DS FRAMEWORK PRECISION MEASUREMENT

Precision is related to the refinement of the measuring process. It is concerned with how small a difference the measuring device can detect (Pearson, 2010). In this study, we tested the precision of the framework by applying it to real life projects that have already been implemented in the software development environment. A successful study is one that demonstrates high precision and high accuracy.

As mentioned in the methodology, in this study we used 15% of the collected data for precision testing purposes. We did this by comparing the results of the decision support framework against the actual software methodology that was adopted for the project. In this study we employed the precision testing formula by (Jizba 2000). Jizba explains precision as the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. Jibza defines the formula for precision as A/A+C * 100/1; where A is true/positive and C is false/negative.

Figure 21 is the representation of the testing data. As seen below, 15 projects were used for testing the precision of the decision support framework. 12 projects recorded positive correspondence between the DS framework results and the actual applied software development methodology. Furthermore, 3 projects recorded non-correspondence to the DS framework results. This concludes that the decision support framework is 80% precise.

Precision: 12/ (12+3) * 100 = 80%.



	Project Name	. On a scale of 1 to 5, were the req clear and documented?	On a scale of 1 to 5, how big was the project scope?	On a scale of 1 to 5, how complex was the project?	On a scale of 1 to 5, does this project have high or low user impact?	On a scale of 1 to 5, how long is the duration of this project ? Short or long?	Was the project succesful?	Methodology Applied	DSS Framework recommendation	DSS Accurate? Yes/ No?
1	NPG Decision Support System	4	2	2	3	2	Yes	Waterfall	Waterfall	Yes
2	Enhancement of CRM	2	2	4	4	2	No	Agile	Agile	Yes
3	Health Information System	2	3	5	4	3	Yes	Agile	Agile	Yes
4	Development of FinSocial Mobile Appication	4	2	3	5	3	No	Agile	Spiral	No
5	Information Analysis and Forensic Tool	2	4	5	2	2	Yes	RAD	RAD	Yes
6	Surveilance Setup and Config	1	2	2	4	2	Yes	Prototyping	Prototyping	Yes
7	Development of Encryption and Decryption	5	3	5	1	2	Yes	Hybrid	Spiral	No
8	New intranet	2	2	1	5	2	Yes	Prototype	Prototyping	Yes
9	New Website	2	2	1	5	2	Yes	Prototype	Prototyping	Yes
10	SharePoint Upgrade	1	2	4	4	2	Yes	JAD	JAD	Yes
11	Implementation of Learner Management System	5	2	3	5	3	Yes	Hybrid	Hybrid	Yes
13	Management and Learning Management Solutions	4	2	2	3	2	Yes	Hybrid	Waterfall	No
14	Upgrade Accreditation/Reaccreditation System	4	2	2	3	2	Yes	Waterfall	Waterfall	Yes
15	Develop mobile app for Enforcement Management System	2	4	4	5	3	Yes	Prototype	Prototyping	Yes

Figure 21: DS Framework Precision Test Results

4.4 INTERPRETATION AND DISCUSSION OF FINDINGS

From the analysis reported above, the research question that proposes the extent to which project characteristics inform the decision for adoption of software development methodologies was supported, the DSS framework was built using SQL programming language and the results of the formulated DSS framework are in agreement with past studies. Mahapatra (2015), described a comparative approach among different software development methodologies in order to be able to select the most appropriate methodology. In similar vein, Munassar and Govardhan (2010) presented different models of software development agreeing with Mahapatra (2015) theory for SDM selection.

Furthermore, in the findings of this study, the DSS framework proves that the waterfall model should be applied where requirements are well documented, clear and fixed. Besides, the findings reveal factors that need to be considered when adopting SDM. These factors are: product definition need to be stable, technology is understood and is not complex, project is short, and there are ample resources with required expertise available to support the product. This findings corroborated with the findings of past studies (Despa, 2014:55; Sheffield & Lemétayer, 2013; Vijayasarathy & Butler, 2016). For example, Despa (2014:55) found that, to choose appropriate software methodology, it is important to consider certain factors such as the developer's technical expertise and project complexity. Likewise, Sheffield and Lemétayer (2013) reported that a project factor, the empowerment of the project team, is a significant factor for choosing appropriate software methodologies. According to Vijayasarathy and Butler (2016), project characteristic is important in choosing software methodologies today because project managers and other team members reported that traditional methodologies, like waterfall model, are still popular and are in use today due to certain organizational, project, and team characteristics.

Although, the current study results was in contrast with findings of Victor Szalvay (2008), who reported that traditional development methodologies emphasize that complex systems can be built in a single pass without necessarily revisiting requirements in light of changing high-technology environment and business needs. One possible reason for this contrary result may be linked to a need for a perfect knowledge of the client's needs, according to Paul (2008) this is a vital factor which is not the case in todays' complex and high-technology environment of the twenty-first century (SEI, 2006).

Additionally, the DSS results are in agreement that the agile and scrum methodology should be applied in short term projects, where there is high customer interaction and also where requirements are not fixed. This findings offer credence to previous studies (Abrahamsson *et al.*, 2017b; Despa, 2014; Dybå & Dingsøyr, 2008). These authors stressed that for a methodology to be considered as agile it has to possess features like being cooperative (i.e. stallholders working constantly together with close communication), the software development must be straightforward (easy to learn and to modify, well documented), and it has to be adaptive (flexible enough to make last moment changes). According to Dybå and Dingsøyr (2008) Agile methodologies do not follow stringent process, progress is dependable on project group understanding and development, which is relevant to promote early delivery of working code, and involvement of project owner (Despa, 2014).

An advantage of the result of the present study is that the DSS framework will help to promote and advocate for quick response to changes in the environment, user requirement and deadlines. Likewise, the DSS results supports the core values and principles that define an agile world view (Cohen et al.,2004; Dybå & Dingsøyr, 2008), especially the core values of customer collaboration rather than contract negotiation and responding to change instead of following a plan.

In addition, the present findings of the DSS, which support high customer interaction with requirements not being fixed, has a contributing factor to the IT sector. This is because the framework will guide in the adoption of appropriate software development methodologies that would fit each organization and project setting and further help in reducing the challenges faced by today's software professionals. Scholars (Dingsøyr et al. 2018; Zykov, 2018) have noted that developing software which is within cost, schedule, fulfils customer requirements and is reliable, seems to be the ultimate challenge in IT industries.

The above contribution will explain why Abrahamsson et al. (2017:106) recently stated that introduction of several software methodologies is not the solution to addressing software crisis but developing a functional decision framework will alleviate such crisis. In the view of Abrahamsson *et al.* (2017:106) the rather frequent release of new agile methods into the market will bring about confusion instead of clarity. Hence, the findings of the DSS is relevant to influence the selection of appropriate methodologies thereby enabling companies within the software development industry, particularly in South Africa, to deliver projects successfully.

Additionally, the DSS also found that the RAD methodology should be applied for complex systems as it uses automated code generating tools. The framework also agrees that RAD should be chosen only if business interaction is present and can provide knowledge. Similarly as with Agile, RAD welcomes changes in requirements. This finding confirms findings of Geambaşu *et al.* (2011:491) who in analysing RAD and other agile methodologies identified system complexity as among the ten factors that influence the decision of choosing the most adequate development methodology for a specific project,

The DSS results supports the theory that prototyping methodology should be applied where requirements are not clear and also where the system requires high level of customer interaction. The results prove that this methodology can be applied for simple projects, however the projects tend to become complex as iteration occurs. These findings is consistent with the theory of Abrahamsson *et al.* (2017b), who claim that it is significant for software development to be incremental in nature, and to be cooperative (stallholders working constantly together with close communication). This will explain why Harb *et al.* (2015), in the United States, acknowledges that for a project to be successful in choosing an appropriate software methodology, it requires a multi-criteria decision approach.

Further findings of the DSS results also prove that the spiral methodology can be applied to large and risky projects that have complex requirements. The spiral methodology is a blend of waterfall and prototyping with added risk analysis capability. These findings agrees with Gill et al. (2018) who suggested that traditional plan-driven software development practices such as waterfall and spiral methodology may be more applicable in large projects.

Possible explanations of these findings of the DSS may be associated with the fact that spiral methodology, as a traditional methodology, is predictive, process-focussed as well as document and plan driven (Boehm, 1988; Despa, 2014). Besides, it follow sequential steps, as its core principle (Paul et al., 2008). Likewise, such a methodology include extensive planning, formal processes, comprehensive documentation, and a long-term design process (Despa, 2014).

Finally the DSS results also agree that the hybrid methodology can be applied in project that requires both agile and waterfall approaches. This means that well documented small projects, which have high customer engagement, can use this methodology to deliver their project in shorter, more frequent cycles. These findings has implication for minimizing software crisis (Sharon et al., 2010), moreover, the findings of the present study is relevant as it will equip project managers with the knowledge to choose appropriate SDMs, which in turn will assist to reduce the increasing complexity of software development management today (Despa, 2014) for many Information technology (IT) industries in the world, and particularly within South Africa (Dingsøyr et al., 2018; Silberberg & Africa, 2006).

The aforementioned implication therefore will decrease the complexity and challenges faced by managers in deciding the most appropriate software methodology to adopt in a software project. This is important for IT industries worldwide, because, wrong choice of methodologies have been documented (Overhage et al. 2011; Mahapatra, 2015) to be costly for the organization as it may impact on deliveries, maintenance costs, project budget and reliability. With the market pressure emphasizing the need to issue products faster and faster, the IT industry is forced to release programs/applications as early and frequently as possible, even if the only version available is limited, this has led to the development of more agile software methodologies. As such, the findings of the present study is significantly beneficial and fit to be adopted as an evolutionary approach, which should be understood as a decision support framework for the adoption of software methodologies particularly for within the software development industry in South Africa.

Further findings shows that the factors that affect software methodology adoption are not influenced by the environment. The DSS framework can thus be applied in other areas outside South Africa. The implication of these findings is that the DSS framework is relevant for adoption not only in South Africa, where the study was conducted, but relevant for use worldwide, which makes the proposed DSS framework a good framework for the selection of appropriate SDM for all IT organization. These findings support that of past studies (Clarke & O'Connor, 2012; Ezeh & Anthony, 2013; Kumaresan & Kumar, 2018). These authors presented other organizational factors to be responsible for software project success.

The possible explanation that can account for the findings of the present study can be explained through the argument of Kumaresan and Kumar (2018), who found factors such as organizational, technical, people, and culture to be more important determinant of project success. Additional explanation of the current study finding can be associated with the findings of Clarke and O'Connor (2012) who advocated for situational factors affecting software development process. According to Clarke and O'Connor (2012) factors like nature of the application(s) under development, team size, requirements volatility and personnel experience were acknowledged to be factors affecting software development process. In their views, aligning with the findings of this present study, environment factors are not as crucial as situational factors in predicting the choice of software development methodologies.

4.5 SUMMARY OF THE CHAPTER

This chapter has presented and discussed the findings obtained from the quantitative questionnaire responses that were derived from ninety participants from ten software development companies within Gauteng province in South Africa. While the study purported to investigate the decision support framework for the adoption of software development methodologies in South Africa, the findings obtained supported extant literature for choosing the best fit Software Development Methodologies (SDMs).

In addition, findings of the present study contribute to reducing the software crisis, decrease challenge faced by today's software professionals and importantly, the DSS results is expected to assist in the adoption of appropriate software development methodologies that would fit each organization and project setting particularly in South Africa. Thus, the DSS results present a framework relevant for methodology selection in IT organization.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 INTRODUCTION

This chapter concludes this study with a summary of the key findings and qualified responses to the initial broad research questions. It further acknowledges the limitation of this study and suggests recommendations for future research.

5.2 CONCLUSION

Based on the formulated sub questions posed in the study, the following conclusions were derived:

'How to propose a suitable tool for the selecting the most appropriate methodology to adopt in a software project'.

In this study, this question was answered by proposing a decision support framework to assist organisations and individuals in determining the best methodology to adopt in a software project. The findings of the present study showed that the results of the formulated DS framework is in agreement with extant literature on the theory for SDM selection/adoption, and this finding contribute to existing knowledge that there is no one size fit all methodology.

'To what extent do project characteristics inform the adoption of the most appropriate software development methodology?'.

In this study this sub question was answered by studying existing literature to determine the characteristics which inform the selection of software development methodologies. Furthermore, a research was carried out to determine these effects within the Gauteng province setting. The findings were presented in chapter 4 of this document.

'How to measure the effectiveness of the proposed decision support framework for the adoption of software development methodologies'.

In this study this sub question was answered by performing an evaluation test on the proposed decision support framework. The framework was tested for accuracy and precision using the precision formula as defined by (Jizba 2000). The precision test produced positive results and is also in agreement with extent literature. These findings were presented in chapter 4 of this document.

In summary, the results of the study provide some compelling insights on the adoption of software development methodologies. The objectives of the research were successfully achieved and the main research question was adequately answered by each sub question.

The results of this study prove that the waterfall model should be applied where requirements are well documented, clear and fixed, product definition is stable, technology is understood and is not complex, and where the project is short and there are ample resources with required expertise are available to support the product.

Further findings showed that the DS framework indicated that the agile and scrum methodology should be applied in short term projects, where there is high customer interaction and also where requirements are not fixed.

Moreover, the DS also found that the RAD methodology should be applied for complex systems as it uses automated code generating tools. Besides, the framework submits that RAD should be chosen only if business interaction is present and can provide knowledge. Similarly than with Zgile, RAD welcomes changes in requirements.

The findings of DS supports the theory that prototyping methodology should be applied where requirements are not clear and also where the system requires a high level of customer interaction. The results prove that this methodology can be applied for simple projects, however the projects tend to become complex as iteration occurs.

Additional findings from the DS framework prove that the spiral methodology can be applied to large and risky projects that have complex requirements. The spiral methodology is a blend of waterfall and prototyping with added risk analysis capability.

The DS results also agree that the hybrid methodology can be applied in project that requires both agile and waterfall approaches. This means that well documented small projects, which have high customer engagement, can use this methodology to deliver their project in shorter frequent cycles.

Finally, the findings of the current study show that the DS framework is not environment specific; this is because the factors that affect software methodology adoption are not influenced by the environment, making the DSS framework universally relevant.

5.3 RECOMMENDATION

Based on the study findings mentioned above, it is recommended that IT organizations should adopt the DS framework in the selection/adoption of SDM, this is recommended firstly because developing software which is within cost, schedule, fulfils customer requirements and is reliable will be attainable. Secondly, the developed framework will guarantee the selection of appropriate software development methodologies that would fit each organization and project setting which have tormented the IT industries for decades.

Besides, it is recommended that IT entrepreneurs should make provisions for periodical training that is based on selection of appropriate methodologies, especially for Project Managers. This is important to create awareness, educate Project Managers and further inform them on the most appropriate methodologies to choose for applicable projects. The DSS results have indicated which SDM need to be applied with regards to requirements, systems and projects (simple or complex).

Finally, there is the need to organise internationally conferences that will be aimed at bringing IT organizations together biannually. This is relevant to give progress reports and provide feedbacks as to the most appropriate SDM for different projects in different IT organizations.

5.4. LIMITATION OF THE STUDY

The following have been acknowledged to be study limitations. The methodology used in the current study might have affected study findings, it is suggested that future study should consider using a mixed method approach because this will provide richer information. In addition to this, all measures were based on self-reports, which might have affected the data. This may lead to an increase in the level of common method bias (Conway & Lance, 2010). In addition to the measure used in the study, there is a need for future researchers to combine other measures such as focus group discussion and interviews along with self-reporting measures in eliciting information from Project Managers and staff related working closely with Project Managers.

Lastly, the present study featured samples from only one province (Gauteng province). Should the DS framework be more fully validated, it will require analysis among more provinces. It is suggested that future works should attempt to include more provinces as this might increase sample size, although this is not to say that the results obtained in this study are not generalizable. Regardless of these limitations however, this study is unique, because of the attempt to address decades of challenges faced in the IT organization as regards selecting the best fit SDM necessary to meet time, budget and functional requirements (Marques, Costa, Silva, & Gonçalves, 2017). Thus, it is believed that the study constitutes a genuine contribution to IT literature.

5.5 SUGGESTION FOR FUTURE RESEARCH

It is difficult for a study of this magnitude to be all encompassing, therefore, there is the need for future studies to review decision support framework for the adoption of software development methodologies using mixed method approach so as to explore more into participants understanding, experiences and views on adoption of most appropriate SDM.

It is also suggested that future research should look more into factors that influence choice of SDM. Likewise, there is a need for future studies to focus more on how complex systems can be built.

Finally, futuristic research should address similar studies, using more samples, this can be achieved by extending research setting to include more provinces in the country. This will make findings more generalizable.

REFERENCES

Alashqur, A. (2016). Towards A Broader Adoption Of Agile Software Development Methods. International Journal Of Advanced Computer Science And Applications. 7(12). P.94–98.

Awad, M. 2005. A Comparison Between Agile And Traditional Software Development Methodologies. University Of Western Australia.

Babbie, E. K. 2007. The Practice Of Social Research: Internal Student Edition. 11th Ed. Ca: Wadsworth.

Bassil, Y. 2012. A Simulation Model For The Waterfall Software Development Life Cycle. Arxiv Preprint Arxiv:1205.6904.

Bern, A., Pasi, S. J. A., Nikula, U. & Smolander, K. (2007). Contextual Factors Affecting The Software Development Process – An Initial View. 2nd Ais Sigsand European Symposium On Systems Analysis And Design.

Campanelli, A. & Parreiras, F. (2012). A Conceptual Model For Agile Practices Adoption. Zenodo.Org, 10. Retrieved From Https://Zenodo.Org/Record/12306/Files/A_Conceptual_Model_For_Agile_Practices_Adoption.Pdf

Centers For Medicare & Medicaid Services. (2008). Selecting A Development Approach. Centers For Medicare & Medicaid Services, 1–10. Retrieved From Http://Www.Cms.Gov/Research-Statistics-Data-And-Systems/Cms-Information-Technology/Xlc/Downloads/Selectingdevelopmentapproach.Pdf

Choudhary, Abhishek, Deepak Kasgar, And Lokesh Kashyap. 2015. "Evolvea Frameworkfor Selectingprime Software Developmentprocess." 5:20–24.

Cockburn, A. (2000). Selecting A Project's Methodology. Ieee Software. 17(4). P.64–71. Https://Doi.Org/10.1109/52.854070

Creswell, J.W. (2007). Qualitative Inquiry And Research Design: Choosing Among Five Approaches (2nd Ed.). Thousand Oaks, Ca: Sage.

Dr Winston W Royce. 2016. "Sep 28 2016." The Mississippi Supreme Court 933(August):1–9.

Duggal, P. (2006). Guidelines To Support Choice Of Development Methodology Pravin Duggal Information Systems Bsc. P.1–93.

Flora, H. K., Chande, S. V. & Wang, X. (2014). Adopting An Agile Approach For The Development Of Mobile Applications. International Journal Of Computer Applications. 94(17). P.43–50. Https://Doi.Org/10.5120/16454-6199

Geambaşu, C. V., Jianu, I., Jianu, I. & Gavrilă, A. (2011). Influence Factors For The Choice Of A Software Development Methodology. Accounting And Management Information Systems., 10(4). P.479–494.

Khan, P. M. & Beg, M. M. S. S. (2013). Extended Decision Support Matrix For Selection Of Sdlc-Models On Traditional And Agile Software Development Projects. International Conference On Advanced Computing And Communication Technologies. Acct. 3(1). P.8–15. Https://Doi.Org/10.1109/Acct.2013.12

Liviu Despa, M. (2014). Comparative Study On Software Development Methodologies. Database Systems Journal. 5(3). P.37–56. Https://Doi.Org/10.1109/Mahc.1983.10102

Mahanti, R., Neogi, M. S. & Bhattacherjee, V. (2012). Factors Affecting The Choice Of Software Life Cycle Models In The Software Industry-An Empirical Study. Journal Of Computer Science. 8(8). P.1253–1262.

Mahapatra, H. B. (2015). Selection Of Software Development Methodology (Sdm): A Comparative Approach. International Journal Of Advanced Research In Computer Science And Software Engineering. 5(3). P.58–61.

Manawadu, C. D., Johar, G. & Perera, S. S. N. (2013). An Evaluation Of Software Development Methodology Adoption By Software Developer In Sri Lanka. International Journal Of Computational Engineering Research. 3(11). P.1–11.

Mohammed, N., Munassar, A. & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. International Journal Of Computer Science. 7(5). P.94–101. Https://Doi.Org/10.1.1.403.3201

Orb, A., Eisenhauer, L. & Wynaden, D. (2000). Ethics In Qualitative Research. Journal Of Nursing Scholarship. 33(1). P.93–96. Https://Doi.Org/10.1111/J.1547-5069.2001.00093.X

Pearson. (2010). Research Methods For Sport And Exercise Sciences. P.1–34.

Ramnath, V. (2010). The Level Of Adoption And Effectiveness Of Software Development Methodologies In The Software Development Industry In South Africa. (November). P.80.

Sei. (2006). Cmmi For Development, Version 2.1. Pittsburgh: Carnegie Mellon University. Available From: Http://Www.Sei.Cmu.Edu/Reports/06tr008.Pdf (Accessed 01/10/2018)

Sharma, P. & Singh, D. (2015). Comparative Study Of Various Sdlc Models On Different Parameters. International Journal Of Engineering Research. 4(4). P.188–191. Retrieved From Http://Www.Ijer.In/Ijer/Publication/V4s4/Ijer_2015_405.Pdf

Shekapure, P. S. & Nagar, K. (2015). Problem Solving Using Case Based Reasoning Methodology. 1(11). P.881–887. Https://Doi.Org/01.0401/Ijaict.2015.11.16

Sylvester, I. 2014. "An Overview Of The Development Principles , Stages And Building Blocks Of Expert System." 11(1):44–58.

Verma, J., Bansal, S. & Pandey, H. (2014). Develop Framework For Selecting Best Software Development Methodology. 5(4). P.1067–1070.

Boehm, B.W. 1988. A Spiral Model Of Software Development And Enhancement. *Computer*, 21(5):61-72. Leau,

Chilisa, B. (2011). Indigenous Research Methodologies. Thousand Oaks: Sage.

Chukwuere, J. E. 2016. Toward A Culture-Oriented E-Learning System Development Framework (E-Lsdf) In Higher Education Institutions: South Africa. Mafikeng: North-West University (Doctor Information Systems).

Clarke, P. & O'connor, R.V. 2012. The Situational Factors That Affect The Software Development Process: Towards A Comprehensive Reference Framework. *Information And Software Technology*, 54(5):433-447.

Cockburn, A. 2000. Selecting A Project's Methodology. *Ieee Software*, 17(4):64-71.
Cohen, D., Lindvall, M. & Costa, P. 2004. An Introduction To Agile Methods. *Advances In Computers*, 62(03):1-66.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F.J. & Tovar, E. 2014. Project Managers In Global Software Development Teams: A Study Of The Effects On Productivity And Performance. *Software Quality Journal*, 22(1):3-19.

Creswell, J. W. 2014. Research Design: Qualitative, Quantitative, And Mixed Methods Approaches. Fourth Edition. Thousand Oaks, Ca: Sage Publications, Inc.

Creswell, J. W. 2008. Educational Research: Planning, Conducting, And Evaluating Quantitative And Qualitative Approaches To Research, 3rd Ed. Upper Saddleriver, Nj: Merill/Perason Education.

Crotty, M. (1998). The Foundations Of Social Research: Meaning And Perspective In The Research Process. London: Sage

De Vasconcelos, J.B., Kimble, C., Carreteiro, P. & Rocha, Á. 2017. The Application Of Knowledge Management To Software Evolution. *International Journal Of Information Management*, 37(1):1499-1506.

Delcheva, Y. 2018. Challenges During The Transition To Agile Methodologies: A Holistic Overview.

Despa, M.L. 2014. Comparative Study On Software Development Methodologies. *Database Systems Journal*, 5(3):37-56.

Dingsøyr, T., Moe, N.B., Fægri, T.E. & Seim, E.A. 2018. Exploring Software Development At The Very Large-Scale: A Revelatory Case Study And Research Agenda For Agile Method Adaptation. *Empirical Software Engineering*, 23(1):490-520.
Dybå, T. & Dingsøyr, T. 2008. Empirical Studies Of Agile Software Development: A Systematic Review. *Information And Software Technology*, 50(9-10):833-859.

Elliot, D. J. 2002. Philosophical Perspectives On Research. (In Colwell, R. & Richardson, C. (Eds.). The New Handbook On Research Music Teaching And Learning (Pp. 85-102). Oxford: Oxford University Press).

Emekako, R. U. 2015. Management Strategies For Learner Discipline In Secondary Schools In Ngaka Modiri-Molema District Of The North West Province. Mafikeng: North-West University (Masters Education Management).

Erickson, J., Lyytinen, K. & Siau, K. 2005. Agile Modeling, Agile Software Development, And Extreme Programming: The State Of Research. *Journal Of Database Management (Jdm)*, 16(4):88-100.

Ezeh, A. & Anthony, P. 2013. Factors Influencing Knowledge Sharing In Software Development: A Case Study At Volvo Cars It Torslanda.

Farrell, A. 2007. Selecting A Software Development Methodology Based On Organizational Characteristics. *An Essay Submitted In Partial Fulfillment Of The Requirements For The Degree Of "Master Of Science In Information Systems", Athabasca University, Athabasca.*

Fitzgerald, B. 1996. Formalized Systems Development Methodologies: A Critical Perspective. *Information Systems Journal*, 6(1):3-23.

Geambaşu, C.V., Jianu, I., Jianu, I. & Gavrilă, A. 2011. Influence Factors For The Choice Of A Software Development Methodology. *Accounting And Management Information Systems*, 10(4):479-494.

Georgiadou, E. 2003. Software Process And Product Improvement: A Historical Perspective. *Cybernetics And Systems Analysis*, 39(1):125-142.

Gill, A.Q., Henderson-Sellers, B. & Niazi, M. 2018. Scaling For Agility: A Reference Model For Hybrid Traditional-Agile Software Development Methodologies. *Information Systems Frontiers*, 20(2):315-341.

Goodpasture, J.C. 2010. Project Management The Agile Way: Making It Work In The Enterprise: J. Ross Publishing.

Griffin, A.S. & Brandyberry, A.A. 2010. System Development Methodology Usage In Industry: A Review And Analysis. *Journal Of Information Systems Applied Research*, 3(19):1-18.

Harb, Y., Noteboom, C. & Sarnikar, S. 2015. Evaluating Project Characteristics For Selecting The Best-Fit Agile Software Development Methodology: A Teaching Case. *Journal Of The Midwest Association For Information Systems*, 1:33.
Highsmith, J. & Cockburn, A. 2001. Agile Software Development: The Business Of Innovation. *Computer*, 34(9):120-127.

Highsmith, J.A. & Highsmith, J. 2002. Agile Software Development Ecosystems. Vol. 13: Addison-Wesley Professional.

Jiang, L. & Eberlein, A. 2008. Towards A Framework For Understanding The Relationships Between Classical Software Engineering And Agile Methodologies. (*In*. Proceedings Of The 2008 International Workshop On Scrutinizing Agile Practices Or Shoot-Out At The Agile Corral Organised By: Acm. P. 9-14).

Khan, P. & Beg, M.S. 2013. Extended Decision Support Matrix For Selection Of Sdlc-Models On Traditional And Agile Software Development Projects. (*In*. Advanced Computing And Communication Technologies (Acct), 2013 Third International Conference On Organised By: Ieee. P. 8-15).

Khoza, L.T. & Pretorius, A.B. 2017. Factors Negatively Influencing Knowledge Sharing In Software Development. *South African Journal Of Information Management*, 19(1):1-9.

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Trektere, K., Mc Caffery, F., Vahid, G., Felderer, M., Linssen, O. & Hanser, E. 2018. Hybrid Software Development Approaches In Practice: A European Perspective. *Ieee Software*.

Kumaresan, K. & Kumar, R. 2018. Analysis Of Software Failure Factors And Criteria To Increase The Software Quality. *Paripex-Indian Journal Of Research*, 6(11).

Lalsing, V., Kishnah, S. & Pudaruth, S. 2012. People Factors In Agile Software Development And Project Management. *International Journal Of Software Engineering & Applications*, 3(1):117.

Lee, J.-C., Shiue, Y.-C. & Chen, C.-Y. 2016. Examining The Impacts Of Organizational Culture And Top Management Support Of Knowledge Sharing On The Success Of Software Process Improvement. *Computers In Human Behavior*, 54:462-474.

Leung, H. & Fan, Z. 2002. Software Cost Estimation. Handbook Of Software Engineering And Knowledge Engineering: Volume Ii: Emerging Technologies. World Scientific. P. 307-324).

Maienschein, J. 2002. Competing Epistemologies And Developmental Biology. In Creath, R. & Maienschein, J. (Eds.), Biology And Epistemology (Pp. 127-137). Cambridge, Uk: Cambridge University Press.

Manawadu, C., Johar, M.G.M. & Perera, S. 2013. An Evaluation Of Software Development Methodology Adoption By Software Developer In Sri Lanka. *Editorial Committees*:84.
Maree, K (Ed). 2010. First Steps In Research. Pretoria: Van Schaik.

Marsh, D., And Furlong, P. 2002. A Skin, Not A Sweater: Ontology And Epistemology In Political Science. In Marsh, D & Stoker, G (Eds.), Theory And Methods In Political Science, 2nd Edition: 17-41. New York: Palgrave Macmillan.

Mcmillian, J.H And Schumacher, S. 2010. Research In Education: Evidence-Based Inquiry. 7th Ed. Pearson: Boston, Ma.

Moe, N.B., Aurum, A. & Dybå, T. 2012. Challenges Of Shared Decision-Making: A Multiple Case Study Of Agile Software Development. *Information And Software Technology*, 54(8):853-865.

Moniruzzaman, A. & Hossain, D.S.A. 2013. Comparative Study On Agile Software Development Methodologies. *Arxiv Preprint Arxiv:1307.3356*.

Morley, C., Hugues, J. & Leblanc, B. 2000. Uml Pour L'analyse D'un Système D'information-Le Cahier Des Charges Du Maître D'ouvrage.

Munassar, N.M.A. & Govardhan, A. 2010. A Comparison Between Five Models Of Software Engineering. *International Journal Of Computer Science Issues (Ijcsi)*, 7(5):94.

Novak, J. D. And Cañas, A. J. 2008. The Theory Underlying Concept Maps And How To Construct And Use Them, Technical Report Ihmc Cmaptools. Http://Cmap.Ihmc.Us/Publications/Researchpapers/Theoryunderlyingconceptmaps.Pdf Date Of Access: 20/11/2018.

O'leary, Z. 2010. The Essential Guide To Doing Your Research Project. London: Sage Publications.

Passos, C., Mendonça, M. & Cruzes, D.S. 2014. The Role Of Organizational Culture In Software Development Practices: A Cross-Case Analysis Of Four Software Companies. (*In*. Software Engineering (Sbes), 2014 Brazilian Symposium On Organised By: Ieee. P. 121-130).

Paul, J. 2008. Quantitative Approach For Lightweight Agile Process Assessment. University Of Turku.

Paul, J., Knuutila, T. & Järvi, A. 2008. Quantitative Approach For Lightweight Agile Process Assessment. *Master's Thesis, University Of Turku*, 7.

Rajagopalan, S. & Mathew, S.K. 2016. Choice Of Agile Methodologies In Software Development: A Vendor Perspective. *Journal Of International Technology And Information Management*, 25(1):3.

Ramnath, V. 2010. The Level Of Adoption And Effectiveness Of Software Development Methodologies In The Software Development Industry In South Africa. University Of Pretoria.
Repko, A.F. 2012. Interdisciplinary Research: Process And Theory. Second Ed. La: Sage Publications.

Rising, L. & Janoff, N.S. 2000. The Scrum Software Development Process For Small Teams. *Ieee Software*, 17(4):26-32.

Rosenberg, D. & Stephens, M. 2008. Extreme Programming Refactored: The Case Against Xp: Apress.

Roses, L.K., Windmöller, A. & Carmo, E.A.D. 2016. Favorability Conditions In The Adoption Of Agile Method Practices For Software Development In A Public Banking. *Jistem-Journal Of Information Systems And Technology Management*, 13(3):439-458.

Russo, N., Wynekoop, J. & Walz, D. 1995. The Use And Adaptation Of System Development Methodologies. *Managing Information & Communications In A Changing Global Environment, Idea Group Publishing, Pa*.

Schwaber, K. & Beedle, M. 2002. Agile Software Development With Scrum. Vol. 1: Prentice Hall Upper Saddle River.

Sharon, I., Dos Santos Soares, M., Barjis, J., Van Den Berg, J. & Vrancken, J.L. 2010. A Decision Framework For Selecting A Suitable Software Development Process. (*In*. Iceis (3) Organised By. P. 34-43).

Sheffield, J. & Lemétayer, J. 2013. Factors Associated With The Software Development Agility Of Successful Projects. *International Journal Of Project Management*, 31(3):459-472.

Silberberg, R. & Africa, I. 2006. An Investigation Into Methods Used To Develop Software Systems. *Elektron Journal-South African Institute Of Electrical Engineers*, 23(2):56.

Sommerville, I. 2016. Software Engineering. London Pearson
Tavares, B.G., Da Silva, C.E.S. & De Souza, A.D. 2017. Risk Management Analysis In Scrum Software Projects. *International Transactions In Operational Research*.

Sturgeon, S., Martin, M. G. F., And Grayling, A. C. 1995. Epistemology. (In Grayling, A. C. (Ed.). Philosophy 1: A Guide Through The Subject (P. 7-60). New York: Oxford University Press).

Vijayasarathy, L.R. & Butler, C.W. 2016. Choice Of Software Development Methodologies: Do Organizational, Project, And Team Characteristics Matter? *Ieee Software*, 33(5):86-94.

Vinekar, V., Slinkman, C.W. & Nerur, S. 2006. Can Agile And Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management*, 23(3):31-42.

Vliet, H.V. 2008. Software Engineering : Principles And Practice. 3rd Ed. Chichester, England ;: John Wiley & Sons.

Williams, L. 2010. Agile Software Development Methodologies And Practices. Advances In Computers. Elsevier. P. 1-44).

Xu, P. & Ramesh, B. 2008. Using Process Tailoring To Manage Software Development Challenges. *It Professional*, 10(4).

Zykov, S.V. 2018. Managing Software Crisis: A Smart Way To Enterprise Agility. Cham: Springer International Publishing. Available:

Harma, Sheetal & Sarkar, Darothi & Gupta, Divya. (2012). Agile Processes And Methodologies: A Conceptual Study. International Journal On Computer Science And Engineering. 4.

Jizba, R. 2000. "Measuring Search Effectiveness." *Creighton University Health Sciences Library*.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2017a. Agile Software Development Methods: Review And Analysis. *Arxiv Preprint Arxiv:1709.08439*.

Aughenbaugh, J.M. And Paredis, C.J., 2004, January. The Role And Limitations Of Modeling And Simulation In Systems Design. In *Asme 2004 International Mechanical Engineering Congress And Exposition* (Pp. 13-22). American Society Of Mechanical Engineers.

Awad, M. 2005. A Comparison Between Agile And Traditional Software Development Methodologies. *University Of Western Australia*.

Babbie, E. K. 2007. The Practice Of Social Research: Internal Student Edition. 11th Ed. Ca: Wadsworth.

Bassil, Y. 2012. A Simulation Model For The Waterfall Software Development Life Cycle. *Arxiv Preprint Arxiv:1205.6904*.

Boehm, B.W. 1988. A Spiral Model Of Software Development And Enhancement. *Computer*, 21(5):61-72.

Chilisa, B. (2011). Indigenous Research Methodologies. Thousand Oaks: Sage.

Chukwuere, J. E. 2016. Toward A Culture-Oriented E-Learning System Development Framework (E-Lsdf) In Higher Education Institutions: South Africa. Mafikeng: North-West University (Doctor Information Systems).

Clarke, P. & O'connor, R.V. 2012. The Situational Factors That Affect The Software Development Process: Towards A Comprehensive Reference Framework. *Information And Software Technology*, 54(5):433-447.

Cockburn, A. 2000. Selecting A Project's Methodology. *Ieee Software*, 17(4):64-71.
Cohen, D., Lindvall, M. & Costa, P. 2004. An Introduction To Agile Methods. *Advances In Computers*, 62(03):1-66.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F.J. & Tovar, E. 2014. Project Managers In Global Software Development Teams: A Study Of The Effects On Productivity And Performance. *Software Quality Journal*, 22(1):3-19.
Creswell, J. W. 2014. Research Design: Qualitative, Quantitative, And Mixed Methods Approaches. Fourth Edition. Thousand Oaks, Ca: Sage Publications, Inc.

Creswell, J. W. 2008. Educational Research: Planning, Conducting, And Evaluating Quantitative And Qualitative Approaches To Research, 3rd Ed. Upper Saddleriver, Nj: Merill/Perason Education.

Crotty, M. (1998). The Foundations Of Social Research: Meaning And Perspective In The Research Process. London: Sage

De Vasconcelos, J.B., Kimble, C., Carreteiro, P. & Rocha, Á. 2017. The Application Of Knowledge Management To Software Evolution. *International Journal Of Information Management*, 37(1):1499-1506.

Delcheva, Y. 2018. Challenges During The Transition To Agile Methodologies: A Holistic Overview.

Despa, M.L. 2014. Comparative Study On Software Development Methodologies. *Database Systems Journal*, 5(3):37-56.

Dingsøyr, T., Moe, N.B., Fægri, T.E. & Seim, E.A. 2018. Exploring Software Development At The Very Large-Scale: A Revelatory Case Study And Research Agenda For Agile Method Adaptation. *Empirical Software Engineering*, 23(1):490-520.

Dybå, T. & Dingsøyr, T. 2008. Empirical Studies Of Agile Software Development: A Systematic Review. *Information And Software Technology*, 50(9-10):833-859.

Elliot, D. J. 2002. Philosophical Perspectives On Research. (In Colwell, R. & Richardson, C. (Eds.). The New Handbook On Research Music Teaching And Learning (Pp. 85-102). Oxford: Oxford University Press).

Emekako, R. U. 2015. Management Strategies For Learner Discipline In Secondary Schools In Ngaka Modiri-Molema District Of The North West Province. Mafikeng: North-West University (Masters Education Management).

Erickson, J., Lyytinen, K. & Siau, K. 2005. Agile Modeling, Agile Software Development, And Extreme Programming: The State Of Research. *Journal Of Database Management (Jdm)*, 16(4):88-100.

Ezeh, A. & Anthony, P. 2013. Factors Influencing Knowledge Sharing In Software Development: A Case Study At Volvo Cars It Torslanda.

Farrell, A. 2007. Selecting A Software Development Methodology Based On Organizational Characteristics. *An Essay Submitted In Partial Fulfillment Of The Requirements For The Degree Of "Master Of Science In Information Systems", Athabasca University, Athabasca*.

Fitzgerald, B. 1996. Formalized Systems Development Methodologies: A Critical Perspective. *Information Systems Journal*, 6(1):3-23.
Geambaşu, C.V., Jianu, I., Jianu, I. & Gavrilă, A. 2011. Influence Factors For The Choice Of A Software Development Methodology. *Accounting And Management Information Systems*, 10(4):479-494.

Georgiadou, E. 2003. Software Process And Product Improvement: A Historical Perspective. *Cybernetics And Systems Analysis*, 39(1):125-142.

Gill, A.Q., Henderson-Sellers, B. & Niazi, M. 2018. Scaling For Agility: A Reference Model For Hybrid Traditional-Agile Software Development Methodologies. *Information Systems Frontiers*, 20(2):315-341.

Goodpasture, J.C. 2010. Project Management The Agile Way: Making It Work In The Enterprise: J. Ross Publishing.

Griffin, A.S. & Brandyberry, A.A. 2010. System Development Methodology Usage In Industry: A Review And Analysis. *Journal Of Information Systems Applied Research*, 3(19):1-18.

Harb, Y., Noteboom, C. & Sarnikar, S. 2015. Evaluating Project Characteristics For Selecting The Best-Fit Agile Software Development Methodology: A Teaching Case. *Journal Of The Midwest Association For Information Systems*, 1:33.

Highsmith, J. & Cockburn, A. 2001. Agile Software Development: The Business Of Innovation. *Computer*, 34(9):120-127.

Highsmith, J.A. & Highsmith, J. 2002. Agile Software Development Ecosystems. Vol. 13: Addison-Wesley Professional.

Jiang, L. & Eberlein, A. 2008. Towards A Framework For Understanding The Relationships Between Classical Software Engineering And Agile Methodologies. (*In*. Proceedings Of The 2008 International Workshop On Scrutinizing Agile Practices Or Shoot-Out At The Agile Corral Organised By: Acm. P. 9-14).

Khan, P. & Beg, M.S. 2013. Extended Decision Support Matrix For Selection Of Sdlc-Models On Traditional And Agile Software Development Projects. (*In*. Advanced Computing And Communication Technologies (Acct), 2013 Third International Conference On Organised By: Ieee. P. 8-15).

Khoza, L.T. & Pretorius, A.B. 2017. Factors Negatively Influencing Knowledge Sharing In Software Development. *South African Journal Of Information Management*, 19(1):1-9.

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Trektere, K., Mc Caffery, F., Vahid, G., Felderer, M., Linssen, O. & Hanser, E. 2018. Hybrid Software Development Approaches In Practice: A European Perspective. *Ieee Software*.

Kumaresan, K. & Kumar, R. 2018. Analysis Of Software Failure Factors And Criteria To Increase The Software Quality. *Paripex-Indian Journal Of Research*, 6(11).

Lalsing, V., Kishnah, S. & Pudaruth, S. 2012. People Factors In Agile Software Development And Project Management. *International Journal Of Software Engineering & Applications*, 3(1):117.

Lee, J.-C., Shiue, Y.-C. & Chen, C.-Y. 2016. Examining The Impacts Of Organizational Culture And Top Management Support Of Knowledge Sharing On The Success Of Software Process Improvement. *Computers In Human Behavior*, 54:462-474.

Leung, H. & Fan, Z. 2002. Software Cost Estimation. Handbook Of Software Engineering And Knowledge Engineering: Volume Ii: Emerging Technologies. World Scientific. P. 307-324).

Maienschein, J. 2002. Competing Epistemologies And Developmental Biology. In Creath, R. & Maienschein, J. (Eds.), Biology And Epistemology (Pp. 127-137). Cambridge, Uk: Cambridge University Press.

Manawadu, C., Johar, M.G.M. & Perera, S. 2013. An Evaluation Of Software Development Methodology Adoption By Software Developer In Sri Lanka. *Editorial Committees*:84.
Maree, K (Ed). 2010. First Steps In Research. Pretoria: Van Schaik.

Marsh, D., And Furlong, P. 2002. A Skin, Not A Sweater: Ontology And Epistemology In Political Science. In Marsh, D & Stoker, G (Eds.), Theory And Methods In Political Science, 2nd Edition: 17-41. New York: Palgrave Macmillan.

Mcmillian, J.H And Schumacher, S. 2010. Research In Education: Evidence-Based Inquiry. 7th Ed. Pearson: Boston, Ma.

Moe, N.B., Aurum, A. & Dybå, T. 2012. Challenges Of Shared Decision-Making: A Multiple Case Study Of Agile Software Development. *Information And Software Technology*, 54(8):853-865.

Moniruzzaman, A. & Hossain, D.S.A. 2013. Comparative Study On Agile Software Development Methodologies. *Arxiv Preprint Arxiv:1307.3356*.

Morley, C., Hugues, J. & Leblanc, B. 2000. Uml Pour L'analyse D'un Système D'information-Le Cahier Des Charges Du Maître D'ouvrage.

Munassar, N.M.A. & Govardhan, A. 2010. A Comparison Between Five Models Of Software Engineering. *International Journal Of Computer Science Issues (Ijcsi)*, 7(5):94.

Novak, J. D. And Cañas, A. J. 2008. The Theory Underlying Concept Maps And How To Construct And Use Them, Technical Report Ihmc Cmaptools. Http://Cmap.Ihmc.Us/Publications/Researchpapers/Theoryunderlyingconceptmaps.Pdf Date Of Access: 20/11/2018.

O'leary, Z. 2010. The Essential Guide To Doing Your Research Project. London: Sage Publications.

Passos, C., Mendonça, M. & Cruzes, D.S. 2014. The Role Of Organizational Culture In Software Development Practices: A Cross-Case Analysis Of Four Software Companies. (*In*. Software Engineering (Sbes), 2014 Brazilian Symposium On Organised By: Ieee. P. 121-130).

Paul, J. 2008. Quantitative Approach For Lightweight Agile Process Assessment. University Of Turku.

Paul, J., Knuutila, T. & Järvi, A. 2008. Quantitative Approach For Lightweight Agile Process Assessment. *Master's Thesis, University Of Turku*, 7.

Rajagopalan, S. & Mathew, S.K. 2016. Choice Of Agile Methodologies In Software Development: A Vendor Perspective. *Journal Of International Technology And Information Management*, 25(1):3.

Ramnath, V. 2010. The Level Of Adoption And Effectiveness Of Software Development Methodologies In The Software Development Industry In South Africa. University Of Pretoria.

Repko, A.F. 2012. Interdisciplinary Research: Process And Theory. Second Ed. La: Sage Publications.

Rising, L. & Janoff, N.S. 2000. The Scrum Software Development Process For Small Teams. *Ieee Software*, 17(4):26-32.

Rosenberg, D. & Stephens, M. 2008. Extreme Programming Refactored: The Case Against Xp: Apress.

Roses, L.K., Windmöller, A. & Carmo, E.A.D. 2016. Favorability Conditions In The Adoption Of Agile Method Practices For Software Development In A Public Banking. *Jistem-Journal Of Information Systems And Technology Management*, 13(3):439-458.

Russo, N., Wynekoop, J. & Walz, D. 1995. The Use And Adaptation Of System Development Methodologies. *Managing Information & Communications In A Changing Global Environment, Idea Group Publishing, Pa*.

Schwaber, K. & Beedle, M. 2002. Agile Software Development With Scrum. Vol. 1: Prentice Hall Upper Saddle River.

Sharon, I., Dos Santos Soares, M., Barjis, J., Van Den Berg, J. & Vrancken, J.L. 2010. A Decision Framework For Selecting A Suitable Software Development Process. (*In*. Iceis (3) Organised By. P. 34-43).
Sheffield, J. & Lemétayer, J. 2013. Factors Associated With The Software Development Agility Of Successful Projects. *International Journal Of Project Management*, 31(3):459-472.

Silberberg, R. & Africa, I. 2006. An Investigation Into Methods Used To Develop Software Systems. *Elektron Journal-South African Institute Of Electrical Engineers*, 23(2):56.

Sommerville, I. 2016. Software Engineering. London Pearson

Tavares, B.G., Da Silva, C.E.S. & De Souza, A.D. 2017. Risk Management Analysis In Scrum Software Projects. *International Transactions In Operational Research*.

Sturgeon, S., Martin, M. G. F., And Grayling, A. C. 1995. Epistemology. (In Grayling, A. C. (Ed.). Philosophy 1: A Guide Through The Subject (P. 7-60). New York: Oxford University Press).

Vijayasarathy, L.R. & Butler, C.W. 2016. Choice Of Software Development Methodologies: Do Organizational, Project, And Team Characteristics Matter? *Ieee Software*, 33(5):86-94.

Vinekar, V., Slinkman, C.W. & Nerur, S. 2006. Can Agile And Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management*, 23(3):31-42.

Vliet, H.V. 2008. Software Engineering : Principles And Practice. 3rd Ed. Chichester, England ;: John Wiley & Sons.

Williams, L. 2010. Agile Software Development Methodologies And Practices. Advances In Computers. Elsevier. P. 1-44).

Xu, P. & Ramesh, B. 2008. Using Process Tailoring To Manage Software Development Challenges. *It Professional*, 10(4).

Zykov, S.V. 2018. Managing Software Crisis: A Smart Way To Enterprise Agility. Cham: Springer International Publishing. Available:

Harma, Sheetal & Sarkar, Darothi & Gupta, Divya. (2012). Agile Processes And Methodologies: A Conceptual Study. International Journal On Computer Science And Engineering. 4.

Jizba, R. 2000. "Measuring Search Effectiveness." *Creighton University Health Sciences Library*.

Aamodt, A. And Plaza, E., 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, And System Approaches. *Ai Communications*, *7*(1), Pp.39-59.

Babbie, E. K. 2007. The Practice Of Social Research: Internal Student Edition. 11th Ed. Ca: Wadsworth.
Batarseh, F.A. And Gonzalez, A.J., 2018. Predicting Failures In Agile Software Development Through Data Analytics. *Software Quality Journal*, *26*(1), Pp.49-66.

Beck, K. And Gamma, E., 2000. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Bedoll, R. 2003. A Tail Of Two Projects: How "Agile" Methods Succeeded After "Traditional" Methods Had Failed In A Critical System-Development Project. In *Extreme Programming And Agile Methods - Xp/Agile Universe*. Vol. 2753. 25–34.

Chilisa, B. (2011). Indigenous Research Methodologies. Thousand Oaks: Sage.
Chukwuere, J. E. 2016. Toward A Culture-Oriented E-Learning System Development Framework (E-Lsdf) In Higher Education Institutions: South Africa. Mafikeng: North-West University (Doctor Information Systems).

Cohen, David, Mikael Lindvall, And Patricia Costa. "An Introduction To Agile Methods." *Advances In Computers* 62, No. 03 (2004): 1-66.

Creswell, J. W. 2014. Research Design: Qualitative, Quantitative, And Mixed Methods Approaches. Fourth Edition. Thousand Oaks, Ca: Sage Publications, Inc.

Crotty, M. (1998). The Foundations Of Social Research: Meaning And Perspective In The Research Process. London: Sage

Cusick, J. J., Prasad, A., & Tepfenhart, W. M. (2008). Global Software Development: Origins, Practices, And Directions. Advances In Computers, 74, 201-269. Doi:10.1016/S0065-2458(08)00606-2

Emekako, R. U. 2015. Management Strategies For Learner Discipline In Secondary Schools In Ngaka Modiri-Molema District Of The North West Province. Mafikeng: North-West University (Masters Education Management).

Elliot, D. J. 2002. Philosophical Perspectives On Research. (In Colwell, R. & Richardson, C. (Eds.). The New Handbook On Research Music Teaching And Learning (Pp. 85-102). Oxford: Oxford University Press).

Haigh, T., 2010, June. Crisis, What Crisis? Reconsidering The Software Crisis Of The 1960s And The Origins Of Software Engineering. In *Second Inventing Europe/Tensions Of Europe Conference, Sofia, Bulgaria*.

Laanti, M., Salo, O. & Abrahamsson, P. 2011. Agile Methods Rapidly Replacing Traditional Methods At Nokia: A Survey Of Opinions On Agile Transformation. *Information And Software Technology*. 53(3):276–290.

Lacey, M., 2012. *The Scrum Field Guide: Practical Advice For Your First Year*. Addison-Wesley Professional.
Layman, L., Williams, L. And Cunningham, L., 2004, June. Exploring Extreme Programming In Context: An Industrial Case Study. In *Agile Development Conference* (Pp. 32-41). Ieee.

Leau, Yu Beng, Wooi Khong Loo, Wai Yip Tham, And Soo Fun Tan. "Software Development Life Cycle Agile Vs Traditional Approaches." In *International Conference On Information And Network Technology*, Vol. 37, No. 1, Pp. 162-167. 2012.

Lei, H., Ganjeizadeh, F., Jayachandran, P.K. And Ozcan, P., 2017. A Statistical Analysis Of The Effects Of Scrum And Kanban On Software Development Projects. *Robotics And Computer-Integrated Manufacturing*, *43*, Pp.59-67

Loftus, C. And Ratcliffe, M., 2005. Extreme Programming Promotes Extreme Learning?. *Acm Sigcse Bulletin*, *37*(3), Pp.311-315.

Maienschein, J. 2002. Competing Epistemologies And Developmental Biology. In Creath, R. & Maienschein, J. (Eds.), Biology And Epistemology (Pp. 127-137). Cambridge, Uk: Cambridge University Press.

Maree, K (Ed). 2010. First Steps In Research. Pretoria: Van Schaik.

Marsh, D., And Furlong, P. 2002. A Skin, Not A Sweater: Ontology And Epistemology In Political Science. In Marsh, D & Stoker, G (Eds.), Theory And Methods In Political Science, 2nd Edition: 17-41. New York: Palgrave Macmillan.

Matthews, T. (2002). Phase I – Systems Engineering Management Plan: A Process Review And
Appraisal Of The Systems Engineering Capability For The Florida Department Of Transportation.

Florida Department Of Transportation. [Online] Requested March 2009. Available At Http://Www.Floridaits.Com/Semp/Files/Pdf_Report/030220-Tmi-V2.Pdf

Mcmillian, J.H And Schumacher, S. 2010. Research In Education: Evidence-Based Inquiry. 7th Ed. Pearson: Boston, Ma.

Moe, N.B., Aurum, A. & Dybå, T. 2012. Challenges Of Shared Decision-Making: A Multiple Case Study Of Agile Software Development. *Information And Software Technology*, 54(8):853-865.

Meulendijk, K.L.; Oud, S. (2007). Project Management Is Risicomanagement. *Automatisering Gids*, Vol. 47, 2007.

Münch, J., Armbrust, O., Kowalczyk, M. And Soto, M., 2012. *Software Process Definition And Management*. Springer Science & Business Media.

Murphy, B., Bird, C., Zimmermann, T., Williams, L., Nagappan, N. & Begel, A. 2013. Have Agile Techniques Been The Silver Bullet For Software Development At Microsoft? In *2013 Acm / Ieee International Symposium On Empirical Software Engineering And Measurement*. Ieee. 75–84.

Mnkandla, E. 2009. About Software Engineering Frameworks And Methodologies. In *Africon 2009*. Vol. 1087. Ieee. 1–5.

Novak, J. D. And Cañas, A. J. 2008. The Theory Underlying Concept Maps And How To Construct And Use Them, Technical Report Ihmc Cmaptools. Http://Cmap.Ihmc.Us/Publications/Researchpapers/Theoryunderlyingconceptmaps.Pdf Date Of Access: 20/11/2018.

Parnas, D. L. And P. C. Clements (1986) "A Rational Design Process: How And Why To Fake It", Ieee Transactions Of Software Engineering, Pp. 346-357.

Repko, A.F. 2012. Interdisciplinary Research: Process And Theory. Second Ed. La: Sage Publications.

Palmer, S. R. And J. M. Felsing (2002). A Practical Guide To Feature-Driven Development. Upper Saddle River, Nj, Prentice-Hall Inc.

Paul, J. (2008). Quantitative Approach For Lightweight Agile Process Assessment (Master"S Thesis, University Of Turku). Available From: Http://Www.Johanpaul.Com/Thesis_Johanpaul.Pdf (Accessed 01/11/208)

Poole, C.J., Murphy, T., Huisman, J.W. & Higgins, A. 2001. Extreme Maintenance. In Proceedings Ieee International Conference On Software Maintenance. Icsm 2001. Ieee Comput. Soc. 301–309.
Sei. (2006). Cmmi For Development, Version 2.1. Pittsburgh: Carnegie Mellon University. Available From: Http://Www.Sei.Cmu.Edu/Reports/06tr008.Pdf (Accessed 01/10/2018)

Scharff, C., 2011, May. Guiding Global Software Development Projects Using Scrum And Agile With Quality Assurance. In *2011 24th Ieee-Cs Conference On Software Engineering Education And Training (Csee&T)* (Pp. 274-283). Ieee.

Senapathi, M. & Srinivasan, A. 2011. Understanding Post-Adoptive Agile Usage -- An Exploratory Cross-Case Analysis. In *2011 Agile Conference*. Ieee. 117–126.

Sharp, H. And Robinson, H. (2004) 'An Ethnographic Study Of Xp Practice', Empirical Software Engineering, Vol. 9, No. 4, Pp.353–375.

Sheffield, J. & Lemétayer, J. 2013. Factors Associated With The Software Development Agility Of Successful Projects. *International Journal Of Project Management*, 31(3):459-472.

Steels, L. (1990). The Components Of Expertise, *Ai Magazine*, 11(2):30–49.

Sriram, R. & Mathew, S.K. 2012. Global Software Development Using Agile Methodologies: A Review Of Literature. In *2012 Ieee International Conference On Management Of Innovation & Technology (Icmit)*. Ieee. 389–393.

Sturgeon, S., Martin, M. G. F., And Grayling, A. C. 1995. Epistemology. (In Grayling, A. C. (Ed.). Philosophy 1: A Guide Through The Subject (P. 7-60). New York: Oxford University Press).

O'leary, Z. 2010. The Essential Guide To Doing Your Research Project. London: Sage Publications.

Szalvay, V. (2008). An Introduction To Agile Software Development. Retrieved From: Http://Www.Danube.Com/System/Files/Wp_Intro_To_Agile.Pdf

Truex, D., Baskerville, R. And Travis, J., 2000. Amethodical Systems Development: The Deferred Meaning Of Systems Development Methods. *Accounting, Management And Information Technologies*, *10*(1), Pp.53-79.

Turk, D., France, R. And Rumpe, B., 2014. Limitations Of Agile Software Processes. *Arxiv Preprint Arxiv:1409.6600*.

Overhage, S. Et Al., 2011. What Makes It Personnel Adopt Scrum? A Framework Of Drivers And Inhibitors To Developer Acceptance. In Proceedings Of The 44th Annual Hawaii International Conference On System Sciences. Ieee, Pp. 1–10.

Griffin, A. S. & Brandyberry, A. A. (2010). System Development Methodology Usage In Industry: A Review And Analysis. Available From: Http://Proc.Conisar.Org/2008/3522/Conisar.2008. Keskin, N. And B. Kunte.Pdf (Accessed 01/10/2018)

Nandhakumar, J. & Avison, D.E. (1999). The Fiction Of Methodological Development:A Field Study Of Information Systems Develo

Varajão, J. Et Al., 2014. Failures In Software Project Management - Are We Alone? A Comparison With Construction Industry. International Journal Of Modern Project Management, 2(1), Pp.22–27

Victor Szalvay. 2008. "Introduction To Agile Software Development." 1–24.

Vijayasarathy, L.R. And Butler, C.W., 2016. Choice Of Software Development Methodologies: Do Organizational, Project, And Team Characteristics Matter?. *Ieee Software*, *33*(5), Pp.86-94.

Williams, L. (2007). A Survey Of Agile Development Methodologies. Retrieved From Http://Agile.Csc.Ncsu.Edu/Sematerials/Agilemethods.Pdf