The relationship between agile systems development methodologies and software process improvement models

C Malambo

22639578

Dissertation submitted in partial fulfilment of the requirements for the degree Magister Scientiae in Computer Science at the Potchefstroom Campus of the North-West University

Supervisor: Prof HM Huisman

11 May 2017

ACKNOWLEDGEMENTS

Firstly, I would like to thank The Almighty Lord for the guidance and wisdom bestowed upon me. It is through His Will and Grace that all things are possible, and I am a living testimony. Glory be to God.

Secondly, I want to extend my gratitude to my family for always believing, and giving their unconditional support through the worst of times and the best of times. It is through their prayers, and at times harsh but honest reminders, that I have been able to accomplish my goals in life. I wish to also extend a warm “thank you” to my partner for the love and encouragement. To my closest friends, thank you for always driving each other to achieve beyond our expectations.

A special “thank you” goes to my supervisor, Professor Magda Huisman, who not only guided me as a supervisor but took on the role of a mother, scolding me whenever I strayed off course. *Baie dankie* Prof, for the Whatsapp messages (strong language) - a stern reminder of the opportunity I had. The Scrum meetings we had in your office have paid dividends. You had the patience and belief in my abilities.

I also want to thank The National Research Foundation for the financial assistance rendered, to afford my studies. To my manager and colleagues, “thank you” for the support given at all times.

To everyone else that had an input, I wish to express a sincere “thank you” for all the support throughout my studies. This dissertation is dedicated to the Malambo clan.

ABSTRACT

Background:

Agile systems development methodologies (ASDMs) have become more frequently deployed to develop software products at a faster pace as compared to their traditional counterparts. The nature of their agility and underlying principles and practices ensures that the software product is of a quality standard.

Software Process Improvement Models (SPIMs) aid organizations in improving the process of software development and ultimately improve software quality. However, the relationship (or lack of) between ASDMs and SPIMs within software- developing organizations is little known. A comprehensive understanding of these two methodologies may assist organisations in producing better-quality software, enhanced information technology project management, and help with devising an organized software development framework.

Objectives:

The primary objective of this study is to investigate the relationship between ASDMs and SPIMs. The co-existence of these two aspects of the study may provide an understanding of the relationship, and results could be utilised as a guide for organisations to determine the best combination between process models and system development methodologies that can be implemented to achieve the highest possible level of maturity.

Methods:

The research was conducted by first reviewing existing literature from books, accredited journal entries and other sources of literature. A survey with the aid of a questionnaire was performed between June 2014 and December 2015. In total, 100 questionnaires were collected, and statistical analysis was performed on the data.

Results:

The study identified five agile methodologies and three different software-process improvement models. Each was subjected to an in-depth literature review. In addition, data concerning the use of agile methodologies and software process models was

collected from information technology professionals. The questionnaire gathered data on the respondent's job category, their organisations' sizes, and the outcomes of the last project in which they had participated.

Conclusion:
The study addressed the industry use of ASDMs and SPIMs, and the interrelations between them. Literature suggests that these two can co-exist and when used together could present greater benefit than when implemented in isolation. CMMi was the most combined process model, with various agile methods such as XP and Scrum. However, in practice, the situation seemed different. The results showed that combinations were rare.

Key Terms: Agile systems development methodologies, relationship, Software Process Improvement models, survey, organisation, Process, Procedure, Project success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS	
ASD	Adaptive Software Development
ASDMs	Agile Systems Development Methodologies/Methodology
AUP	Agile Unified Process
CMM	Capability Maturity Model
CMMi	Capability Maturity Model Integration
DSDM	Dynamic Systems Development Methodology
FDD	Feature Driven Development
IE	Information Engineering
IEC	International Electro-technical Commission
ISO	International Organization for Standardization
LSD	Lean Software Development
PERT	Program Evaluation Review Technique
RAD	Rapid Application Development
RUP	Rational Unified Process
SDLC	System Development Life Cycle
SDMs	Systems Development Methodologies
SEI	Software Engineering Institute
SPICE	Software Process Improvement and Capability Determination
SPI	Software Process Improvement
SPIMs	Software Process Improvement Models
UML	Unified Model Language
XP	Extreme Programming

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In this chapter, the problem statement of the research study will be defined and the objectives of the research study will be addressed. An introduction to Agile Systems Development Methodologies (ASDMs) and Software Process Improvement Models (SPIMs) in general is also presented. The problem to be investigated is whether there exists a relationship between ASDMs and SPIMs. This problem is introduced and the research approach through which the researcher aims to address this problem is discussed. The last section of this chapter outlines the remaining chapters of this study.

1.2 BACKGROUND OF THE STUDY

Agile systems development methodologies have become quite popular and favoured amongst practitioners, compared to the traditional system development methodologies (SDMs) (Iivari & Iivari, 2010; Dybå & Dingsøyr, 2008). However, just like traditional systems development methodologies (SDMs), the challenge critical to organisations is to overcome the resistance to acceptance of agile methodologies. (Chan & Thong, 2009). Why the exceptional popularity of ASDMs, though?

According to Chan and Thong (2008), the emergence of a new generation of systems development methodologies referred to as ‘agile methodologies’ brought about better ways of dealing with today’s dynamic business environment. Traditional SDMs possess some limitations, such as freezing product functionality, their inability to respond to change, less user/customer involvement, and the need to follow a strict plan, are some of the limitations faced with traditional SDMs (Awad, 2005).

Agile systems development methodologies claim to overcome the limitations that arise from traditional plan-driven SDMs (Chan & Thong, 2009). A team of software practitioners published the “Agile Manifesto”, which outlines the principles of agile systems development (Beck *et al.,* 2001). These principles emphasize the importance of individuals and their interactions, customer collaboration, early and continuous delivery of software, and the capability to respond to volatile dynamic requirements.

Iivari and Iivari (2010), define agile system development (based on the concept of

"agility") as the readiness of an information systems development methodology to quickly or inherently create change, be able to proactively or reactively adapt to change, and learn from change – all whilst contributing to perceived customer value in terms of quality of the product, economic benefits, and simplicity of use. Examples of ASDMs include: Agile Unified Process (AUP) (Ambler, 2005), Extreme Programming (XP) (Beck, 1999), Scrum (Highsmith, 2002; Awad, 2005), Dynamic Systems Development Method (DSDM) (Highsmith, 2002), Adaptive Software Development (ASD) (Highsmith, 2000), Crystal (Abrahamsson et al., 2002), Feature-Driven Development (FDD) (Highsmith, 2002), and Lean Software Development (LSD) (Alexandrou, 2011).

The second aspect of this research focuses on Software Process Improvement Models, also known as maturity models. The maturity models are used to develop and refine an organization's software development process. Humphrey (1991) defines the term 'software process' as "the set of activities, methods, and practices that aid developers in the production of software". The most common maturity models found in the industry include models such as Capability Maturity Model Integration (CMMi) (Huang & Han, 2005), the International Organization for Standardization (ISO) series of models (Singels *et al*., 2001), and Software Process Improvement and Capability Determination (SPICE) (Emam et al., 1998).

1.3 PROBLEM STATEMENT

As earlier mentioned, the deployment of ASDMs is becoming more favoured by many organisations. However, "mature" organizations often require process standardization of their software development. There is a need for accreditation, and thus the development process rating becomes a necessity. In order to obtain this accreditation, an organization needs to have an industry-recognized and approved system development methodology in place. The approaches of ASDMs and SPIMs tend to contradict one another, based on the notion that ASDMS are light-weight methodologies that focus on quick delivery of quality products while maturity standards are considered heavy-weight and rigid, as they emphasise quality and documentation (Nguyen, 2010). Theoretically, ASDMs and SPIMs are compatible. However, is it practical for organisations to be agile in their software development approach while following standards?

The main area of conflict between adopting agile processes and maturity models will be

the extent to which the ratings will be affected. Boehm and Turner (2005) believe that agile processes, for instance, are associated with CMMi's Level 5 concept of constantly adapting to improve performance.

Unfortunately, agile methods do not support the degree of documentation and infrastructure needed for lower-level certification. These lower-level requirements may, in fact, make agile methods less effective. However, some agile projects have managed to attain CMMi Level 2 or even 3 grading, though with certain adjustments to the agile methods (Fritzsche & Keil, 2007). Nguyen (2010) investigated the co-existence of agile methods and maturity models. Ambler (2009) conducted a survey on whether a maturity standard (CMMi)-compliant agile process was followed, and 78% of the respondents said that none was followed. Therefore, the question can be posed whether there is a relationship between the use of ASDMs and SPIMs?

1.4 RESEARCH OBJECTIVES

The primary research question of this study is to investigate whether a relationship exists between ASDMs and SPIMs within organizations? In order to answer this research question, the following research objectives will be addressed:

- Objective 1: Study the use and effectiveness of ASDMs in the industry.
- Objective 2: Study the use and effectiveness of SPIMs in the industry.
- Objective 3: Study the relationship between the use of ASDMs and SPIMs.

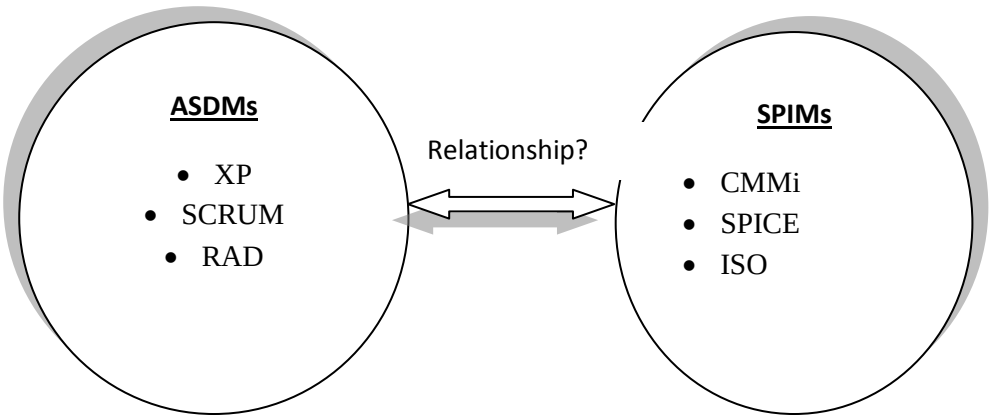

Figure 1: Research objective: Is there Relationship between use of ASDMs and SPIMs?

The two-way relationship between ASDMs and SPIMs, as represented in Figure 1 above, will constitute the main objective of the research to be conducted. The study will also focus on the extent of use of SPIMs and ASDMs in the industry with respect to effectiveness of horizontal and vertical uses within an organization.

1.5 RESEARCH METHODOLOGY

This research is based on a positivist research paradigm. The survey research strategy will be used, as it is ideal to look for patterns and generalizations from sample standardized data of organisations. The use of the survey strategy will aid in unearthing some information with regards to the uses of ASDMs and SPIMs, thus providing the answers to the research-objectives set outlined.

Questionnaires will serve as data-generation methods for this. The use of questionnaires as data-generation methods is frequently associated with survey research strategy, though it is not the only data-generation method associated with surveys (Oates, 2006). The questionnaire for this study will be self-administered. Self-administered questionnaires do not need the presence of the researcher. Some notable advantages for using questionnaires are that they make it possible for information to be collected from a large portion of a population. They also enable respondents to answer predefined questions (Milne, 1999).

Quantitative data will be generated from the questionnaires. The data collected using questionnaires will be analysed by applying available data-analysis tools. Tools such SPSS version 16 will be used to analyse the data. Table 1.1 below summarises the research approach taken for this study.

Table 1.1: Summary of Research Methodology

Research Approach:	**Positivist Research Paradigm**
Research Strategies	Survey
Data Generation Methods	Questionnaires
Data Analysis tools and techniques	**Tools**: Statistical data analysis with SPSS v16 **Techniques**: Crosstabs Correlations Descriptive statistics

1.6 EXPECTED RESEARCH CONTRIBUTION

More organisations are adopting agile methodologies such as Scrum and XP, which are said to be light-weight methods allowing faster production of deliverables. They provide flexibility in hostile, cut-throat environments. At the same time, software organisations wish to be certified with a software process maturity standard such as CMMi, which has structured rigid standards.

However, Boehm and Turner (2003) emphasise that for organisations to be efficient and successful in their software-development process, both the agility gained from using agile methods and the discipline provided by following maturity standards is required. Therefore, both approaches are essential for an organisation to achieve its goals. The

two approaches still contradict one another, as earlier stated, though they are theoretically compatible. An investigation is needed into the relationship of the two approaches in practice. This will help us to answer the question of whether ASDMs are necessary/ adequate to obtain SPIMs certification.

The intended academic contribution of the findings is to extend our body of knowledge on the topic, whilst the practical world can tap into this additional knowledge by helping software developers and organisations that wish to utilise both ASDMs and SPIMs to obtain certification.

1.7 CHAPTER OUTLINE

This section describes the outline of the chapters in this research. Below is a brief description of what the reader of this document will encounter.

Chapter 1: INTRODUCTION

This is an introductory chapter that raises the problem statement upon which the research is based. Thus, the problem statement is centred on the relationship existing between ASDMs and SPIMs, and poses the question of whether ASDMs are necessary/ adequate to obtain SPIMs certification?

Chapter 2: LITERATURE REVIEW

The literature review chapter is comprised of a study of available literature with regards to ASDMs and SPIMs. The study identifies and describes some available ASDMs and SPIMs. The ASDMs to be explored will include: Agile Unified Process (AUP) (Ambler, 2005), Extreme Programming (Beck, 1999), Scrum (Highsmith, 2002; Awad, 2005), Dynamic Systems Development Methodology (DSDM) (Highsmith, 2002), Adaptive Software Development (ASD) (Highsmith, 2000), Crystal (Abrahamsson et al., 2002), Feature Driven Development (FDD) (Highsmith, 2002), and Lean Software Development (LSD) (Alexandrou, 2011).

The maturity standards that will be explored further are: Capability Maturity Model Integration (CMMi) (Huang & Han, 2005), the International Organization for

Standardization (ISO) series of models (Singels *et al*., 2001), and Software Process Improvement and Capability Determination (SPICE) (El Emam et al., 1998).

Chapter 3: RESEARCH METHODOLOGY

This chapter describes the research methodology to be applied to this research. The research methodology selected is based on a positivist research paradigm. The survey will be the research strategy to be used, while data is to be generated with the aid of questionnaires.

Chapter 4: RESULTS OF EMPIRICAL WORK

The outcomes of the research will be reported in this chapter. Analysis and interpretation of data gathered from the questionnaires will give insight into the empirical study conducted. The data collected will be analysed using data-analysis tools and techniques such as SPSS version 16. The results will be presented using statistical analysis through the following tests: frequency tables, crosstabs, correlations, descriptive statistics and factor analysis.

Chapter 5: CONCLUSION AND RECOMMENDATIONS

This chapter serves as the summation of the whole research. In this chapter, literature study findings will be compared to the results obtained from the empirical study. The worth (if any) of the research will be identified together with recommendations for further research in the specific field.

Appendix

This appendix contains an example of the questionnaire that was used during the data-collection process. This questionnaire is divided into four different sections. Each section aims to collect a specific type of data, such as; organisation background information, Systems Methodologies Used, SPIMs Certification attained, and Last Project Outcome.

1.8 SUMMARY

It is evident that software process improvement standards guide organisations on *what-to-do* in their development process; on the other hand, agile methods can be said to provide the *how-to-do* knowledge. This chapter provides a brief introduction to ASDMs and SPIMs. The objective of the study is reviewed in the problem statement. The main objective of the research is to determine the relationship that exists between the use of agile systems development methodologies (ASDMs) and software process improvement models (SPIMs).

To achieve the goals of the research, the researcher has identified and adopted a positivist research paradigm, with survey research strategies and questionnaires being used to gather data from the target population involved in the software-development field.

The following chapter is a literature review to provide insight on ASDMs and SPIMs used in the IT industry.

CHAPTER 2

AGILE SYSTEMS DEVELOPMENT METHODOLOGIES (ASDMS) AND SOFTWARE PROCESS IMPROVEMENT MODELS (SPIMS)

2.1 LITERATURE REVIEW: INTRODUCTION

This chapter discusses the two approaches used by the software development industry. Firstly, agile systems development methodologies (ASDMs) are discussed by first providing a brief background on systems development methodologies in general; and then a selected number of agile methodologies are discussed. Secondly, the other aspect of the study – namely software process improvement models - is introduced and, as with ASDMs, several software process improvement models (SPIMs) are discussed. The last section attempts to find links between the two approaches, as the study looks for relationships between SPIMs and ASDMs.

2.2 SYSTEMS DEVELOPMENT METHODOLOGIES

Agile systems development methodologies have become quite popular and favoured amongst practitioners, when compared to the traditional system development methodologies (SDMs) (Iivari & Iivari, 2010; Dybå & Dingsøyr, 2008). However, just like SDMs, the challenge critical to organisations is to overcome the resistance to acceptance of agile methodologies (Chan & Thong, 2009).

According to Iivari and Iivari (2010), several factors may contribute to the success of agile methods, such as, the “new fashion” that was brought about by early excitement over agile successes. Another reason could be the result of the fundamental changes that have transpired in the systems development terrain.

With all this excitement about Agile Methods doing the rounds, people still don’t fully understand the meaning of the phrase ‘Agile Systems Development Methodology’. The following sections will define a systems development methodology (SDM); define the meaning of agile SDMs; present a history of ASDMs; investigate emerging agile methods; and establish the general effectiveness of ASDMs.

2.2.1 Definition of a Systems Development Methodology

To enhance understanding of the ASDMs, this section provides a broader definition of systems development methodologies. Avison and Fitzgerald (2006) define a systems development methodology as “a collection of procedures, techniques, tools, and documentation aids which help the systems developers in their efforts to implement a new information system”.

However, this research adopts the definition provided by Huisman and Iivari (2006), a “combination of a systems development approach, systems development process model, systems development method and a systems development technique” (Huisman & Iivari, 2006).

A *Systems development approach* is the philosophical aspect upon which a methodology is developed. It includes the guiding principles and beliefs, set of goals, fundamental concepts and principles. For example, object oriented and structured approaches.

A *systems development process model* represents the sequences of steps which a system goes through in its development life-cycle (Wynekoop & Russo, 1993). Some examples of a systems development process models include the iterative, spiral, incremental linear life-cycle, and incremental development process models.

Systems development methods are organized ways of conducting and completing at least a phase of systems development. Methods are comprised of a set of guidelines, activities, tools and techniques to be applied to accomplish the activities, based on a particular philosophy (way of thinking) and the target system (Wynekoop & Russo, 1993). Often the terms method and methodology are used interchangeably. Some examples of a systems development method include Scrum, Extreme Programming and Feature Driven Development.

Systems development techniques are procedures that are to be performed in developing. The procedures may originate from industry-agreed documentation. It requires the use of tools such as prototyping, pair programing and flow diagrams.

2.2.2 Definition of Agile Systems Development Methodologies

The level of agility in Systems Development Methodologies is what defines them. The concept of agility has been extensively analysed by Conboy (2009), who defines agility as:

“The readiness of an information systems development method to quickly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment.’’

Therefore, the agility of an information systems development method is measured on how fast the methodology can create change, anticipate and react to changes in the environment, and provide knowledge in the process whilst maintaining customer satisfaction.

The above definition provides a concise description of the attributes associated with an agile systems development methodology.

2.2.3 Agile Systems Development Methodologies

The emergence of agile systems development methodologies (ASDMs) resulted from the so-called pre-methodology and methodology eras, as described by Avison and Fitzgerald (2006). Systems development projects were ill planned and managed. As a result, system development professionals and academics conceptualized new and better ways of developing systems. They discovered a simplified and human-centred way to quickly develop systems without compromising quality. This new way of thinking brought about the birth of agile systems development methodology, or lightweight methodologies (Avison & Fitzgerald, 2006). Thus, human interaction is one of the fundamental principles of ASDMs. Figure 2.1 shows a timeline of the evolution of agile methodologies and theories. In the next sections, the agile manifesto and, some commonly known types of agile SDMs are discussed.

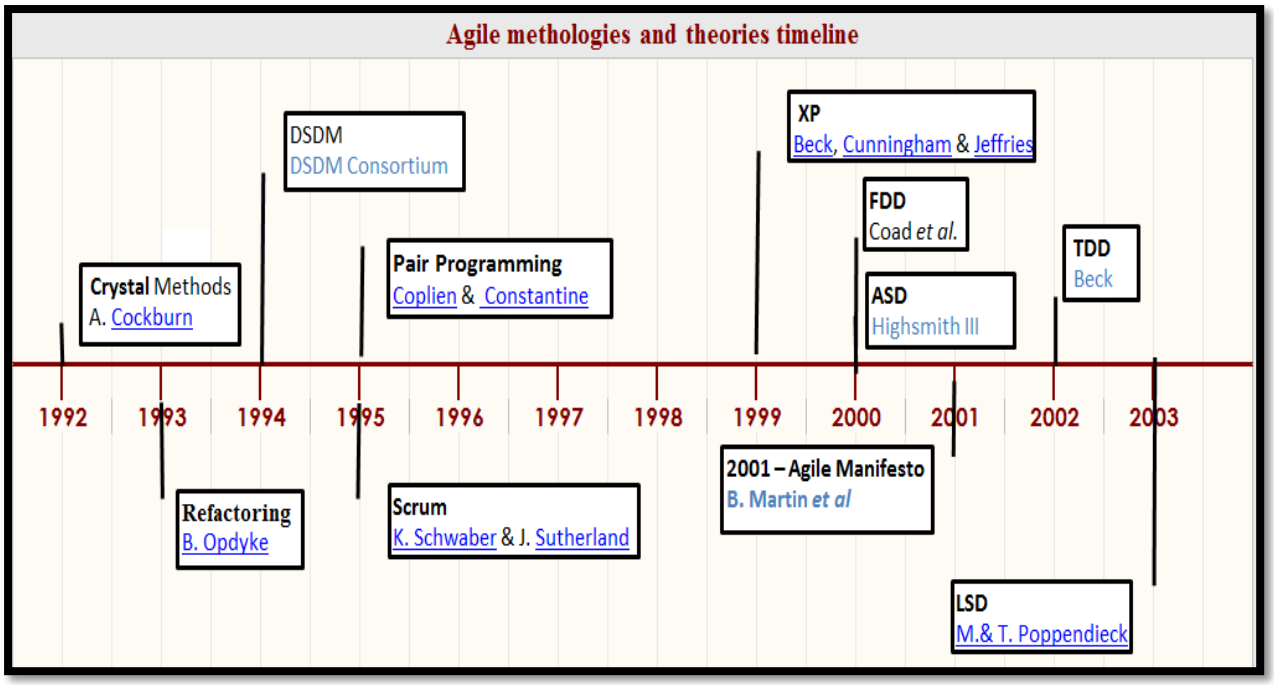

Figure 2.1: Agile Methodologies and Theories timeline

2.2.3.1 Agile Manifesto

The Agile Manifesto for Software Development was drafted to discover better ways of developing software. In drafting the Manifesto, the authors took into consideration certain aspects of software development that they have come to value. These aspects involve valuing:

- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan
- **Individuals and interactions** over processes and tools

The Manifesto emphasises that working software, customer collaboration, responding to change, and individuals and interaction are more valuable to Agile Software Development than having over comprehensive documentations, following plans, negotiating contracts, and processes and tools(BECK et al., 2001).

2.2.3.2 History on the Agile Manifesto

The Agile Manifesto was formed in February 2001 when 17 individuals came together at a ski resort in Utah to ski, relax and try to reach for common ground. Amongst these individuals were preventatives from Crystal Methodologies, Extreme Programming, Feature Driven Development, SCRUM, Adaptive Software Development, DSDM, and other interested parties in finding feasible alternatives to software development processes that that are considered to be heavyweight and documentation-driven.

Alistair Cockburn raised a general concern with the terminology used such as lightweight. However, by the end of the meeting a general consensus had been reached and a new phrase called “agile” was coined. The new phrase of the Manifesto brought satisfaction to the group and was something substantive that they all agreed upon.

As the meeting was coming to an end, Robert Martin, XP mentor joked about making a “mushy” statement. However, the group shared his sentiments that Agile Methodologies are about “mushy” stuff aiming to deliver quality products to their customers. This is done in an environment that promotes collaboration, where people enjoy working with other people that share the same goals and values centred on mutual respect and trust, an environment where people are valued as the most important part of the project rather than just being valued as assets to the organization. The “mushy” stuff basically refers to values and culture.

The outcome of the meeting was the Agile Manifesto. The groups purpose was to uncover better ways of developing software by doing it and assisting others do it. They stated the following values: “We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment” (Highsmith, 2001).

2.2.3.3 Principles of the Agile Manifesto

Beck et al (2001) identifies the following principles as stipulate in the Agile Manifesto: (BECK et al., 2001):

- The main priority is customer satisfaction through the continuous delivery of valuable software.
- Embrace and adapt to changing requirements at any stage in the development

process. This gives the customer a competitive advantage.

- Deliver working software at regular intervals varying in weeks or months, with preference to the shortest time scale.
- Business people and developers must join forces and constantly work together throughout the projects life cycle.
- Reliance on motivated employees and individuals to get the work done, provide them with environment and support they require.
- Enhance personal interactions with staff to convey information to development team.
- Measure progress primarily through working software.
- All participants should be able to maintain a constant pace for an indefinite period of time, keeping in mind that agile processes promote sustainable development.
- Simplicity is of the utmost importance.
- To enhance agility, pay particular attention to technical excellence and good design.
- The best architectures, requirements, and designs are an immediate result of self-organizing teams.
- Regular team meetings are essential for teams to reflect on how they can be more effective, and can be useful to adjust team behaviour accordingly.

2.2.4 Types of ASDMs

This section highlights some well-known and frequently applied ASDMs in the industry. These include Agile Unified Process (AUP) (Ambler, 2005), Extreme Programming (XP) (Beck, 1999), Scrum (Highsmith, 2002; Awad, 2005), Dynamic Systems Development Method/Methodology (DSDM) (Highsmith, 2002), Adaptive Software Development (ASD) (Highsmith, 2000), Crystal family of methodologies (Abrahamsson et al., 2002), Feature Driven Development (FDD) (Highsmith, 2002), and Lean Software Development (LSD) (Alexandrou, 2011), to name but a few.

The establishment of the agile manifesto in 2001 brought extraordinary changes to the field of software manufacturing. The noticeable changes, post-agile manifesto, include: a distinct move towards collaborative development, with people being allowed privileges over processes which previously constrained them. Another was adoption of a “lean” mentality, with the view that unnecessary work may be minimised. Customers/stakeholders began having active roles in shaping the evolution of the end

software product. Lastly, there was acceptance of the occurrence of uncertainties during the process of software development (Dingsøyr et al., 2012).

For the purpose of this research, five of the commonly used ASDMs (XP, FDD, Scrum, LSD and DSDM) are further discussed, based on the definition of a SDM provided in section 1.2 by Huisman and Iivari (2006). This definition is based on the four components upon which an SDM is defined. These four components are:

- Systems development approach;
- Systems development process model;
- Systems development method; and
- Systems development technique

2.2.4.1 Extreme Programming

Kent Beck founded and introduced Extreme Programming (XP) in 1996. XP is moulded on five fundamental values that aim to enhance respect, simplicity, feedback, communication and courage (Jeffries, 2000). XP focuses on delivering immediate benefits to the end-user/customer by reducing the communication gap between the customer and the development team (Abrahamsson & Koskela, 2004).

- ***Systems development approach***

A key pillar upon which XP is built is to recognize, and respect, that people are individuals and, thus, how they think and feel should be taken into consideration. Technically, XP aims to satisfy the customer. Extreme Programming ensures customer satisfaction by delivering working software the customer needs, as they need it. The XP approach empowers system/software developers to confidently respond to changing customer requirements, even when the project is in its later stages of the project life cycle.

- ***Systems development process model***

Extreme Programming follows an iterative and incremental development process model (Jeffries *et al*., 2001). At the end of each XP iteration, the aim is to have working software.

- ***Systems development method***

XP projects undergo a four-phased initial development process that is followed by production support and maintenance until the project ceases (due to various reasons, such as not meeting the new requirements of the business). There are four basic activities/phases that a project goes through, namely Planning and Managing, Designing, Coding, and Testing (Fojtik, 2011). Cohen et al (2004) identify 12 principles/practices of XP. Each practice may affect one or more development phases. Below, the four phases are discussed with the practices that are involved in each phase.

- ***Planning and Managing***

The planning and managing phase involves writing user stories with the aid of story cards. The story cards assist in mapping the project. In this phase, the project is divided into iterations, identification of team members is conducted, and schedules for testing are proposed. The practices use in planning and managing are:

- On-Site Customer: The real-life users/customers are involved in development process. (Affects requirements as the developer may liaise and clarify requirements with customers).
- The Planning Game: This practice is used during collection and clarification of requirements at the start of each iteration. It also addresses maintenance issues in-between iterations.
- Small Releases: Enforces developing team to design-in components and perform tests after each release.

- ***Designing***

This phase makes use of Class, Responsibilities, and Collaboration (CRC) Cards to design the system as a team. CRC cards give code ownership to the entire development team.

This phase and the subsequent phases introduce a term called refactoring. Refactoring describes the continuous simplification of the system without changing its behaviour. Refactoring occurs throughout the project lifecycle. The practices used in designing are:

- Simple Design: XP aims at keeping things as simple as possible for a period. At assists the developers to select a design and conveys how to code sections in cases where there are multiple options.
- System Metaphor: Having a common vision of the project. This provides support for all phases of development by providing a “naming convention” that allow the developing team to be organised. Naming conventions help in overall understanding of the design and reuse code.
- Refactoring: Requires the developer to simplify the design or code if the option is there. Keeping design and code simple facilitates maintenance.
- Coding: During the coding phase, testing is increased so that no mistake slips through the net. The customer is readily available. The iteration schedule is also shortened to one week. During this phase, some ideas might come up on what can still be done to the system.
- Coding Standards: A supporting practice to the collective code ownership practice that aims at keeping code consistent and easy to read and re-factor.
- Pair Programming: A practice that distinguishes XP from other methodologies. A pair of programmers works at the same computer to accomplish certain functionality. A pair of programmers working together may have the same productivity when working individually; however, working as team results in better quality.
- Continuous Integration: An approach to coding that also affects when and how developers integrate new code.
- Collective Code Ownership: A way for code developers to program and give them the option to modify other code.
- 40-Hour Work Week: The developers should not work more than 40 hours a week. This practice relates to management support. (Provides a philosophy on how to manage the work force).
- Testing: In the coding phase, it was noted that, all code must have and pass unit tests can be released. Unit tests are planned from the beginning of the project. In circumstances where a bug is found, tests are created to debug the code. Acceptance tests are run often, and scored against the requirements of the system.
- Unit Testing: A way of coding and a testing approach that provides a test suite against which modifications can be checked.
- Acceptance testing: Tests done at the end of iteration by the customers, to ensure

that all their requirements have been met.

- ***Systems development technique***

XP makes use of pair programming, test driven development (TDD), story cards and continuous integration. Each acceptance test is scored, and the score is then compared with the customers' requirements. If the score is unsatisfactory, immediate changes are made.

2.2.4.1.1 Use of XP

XP works really well in small-to-medium size teams in an environment where there are rapidly changing requirements, or where problems are so complex that they can't be broken down into small parts that can be delivered within a few weeks for each problem (Paulk, 2001).

2.2.4.1.2 Strengths and weaknesses of XP

One of the major strengths of XP is that it can change and adapt to changing and volatile business needs. The cost of adapting to change is minimised, as opposed to conventional programming because the project is broken down into smaller parts. However, some of XP's strengths are also its weaknesses because (Dalalah, 2014):

- The nature of the small team structure means that it will not be possible to tackle projects that require over 20 programmers. This is because 20 is the maximum recommended number of programmers in XP.
- Every problem is broken down into smaller chunks to be solved in a couple of weeks, thus, big problems that cannot be broken down cannot be solved using XP.
- XP thrives on people working closely together under the same roof. Therefore, use of virtual teams will not be beneficial and relocating staff may increase costs.
- Pair programming allows knowledge dissemination between the pair and team. However, if the practice is wrongly implemented, that is, if the pair's skill levels differ, one with high skill level may dominate the partner, resulting in the idleness of the lesser-skilled programmer.

2.2.4.2 Feature Driven Development

Feature Driven Development (FDD) was developed by Jeff De Luca in 1997. The reason behind FDD's development was to meet specific needs of a bank in Singapore. The development team comprised of 50 individuals mandated to finish the project within a year and three months. De Luca's model comprised of five processes namely: Development of an overall model, Build the feature list, planning by feature, Designing by feature, and Building by feature.

FDD is more suitable for new critical projects, projects upgrading and enhancing existing source code, and projects that aim to develop the next version of an existing application (Abrahamsson *et al.,* 2002). The adoption of FDD by an organization is best through a gradual process as the project progresses. Peter Coad was hired to assist with object modelling (Cohen *et al*., 2004). The first process was influenced by Coad's approach to object modelling.

The second process incorporated Coad's concepts of utilizing feature lists to develop tasks and manage functional requirements. The rest of the processes and their integration into a cohesive unit is a result of Jeff De Luca's experience. The project in Singapore was a success, and has led to several implementations of FDD.

Figure 2.2 below represents the meta-process model for these activities. The first three sequential activities establish an overall model shape. The final two activities are iterates for each feature.

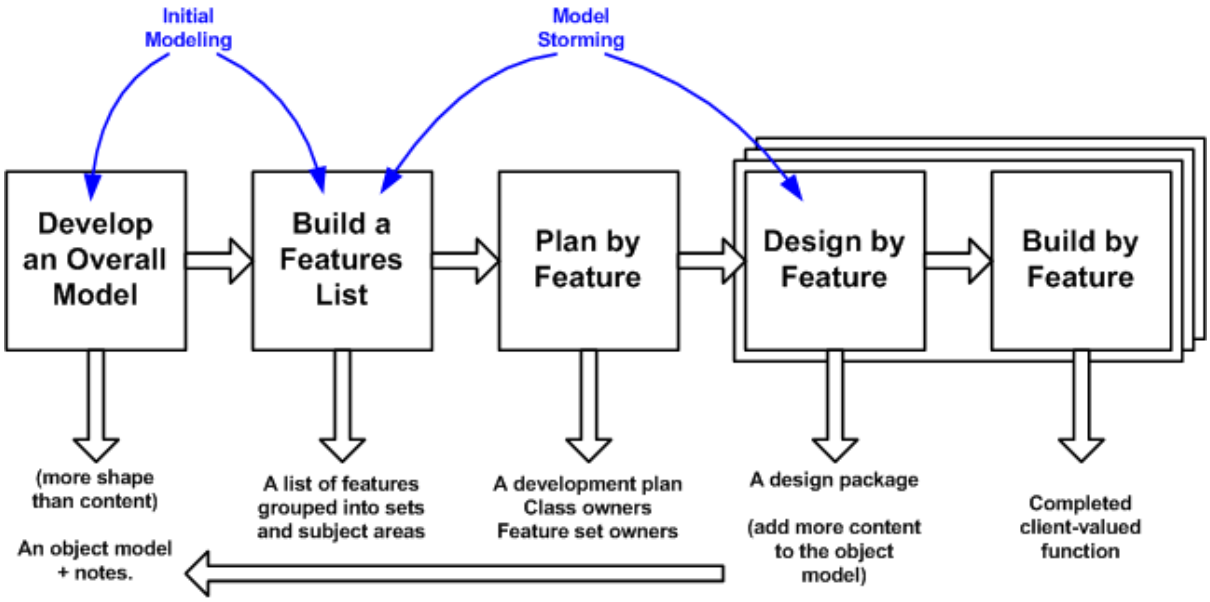

Figure 2.2: FDD Phases (Palmer & Felsing, 2002).

- ***Systems development approach***

FDD follows an adaptive and agile development approach to information systems development (Abrahamsson et al., 2002). However, the FDD approach focuses on the design and building phases rather than covering the entire software-development process. The FDD approach's main purpose is to deliver tangible, working software repeatedly in a timely fashion.

- ***Systems development process model***

The FDD approach follows a model-driven iterative and incremental (Garg, 2009) development process model comprising five sequential activities. However, Palmer and Felsing (2002) state that FDD is also designed to work with other activities of software development, and thus use of a specific process model is not required. The five activities help to accurately monitor the state of the software development project as well as defining targets that mark the progress made on each feature.

- ***Systems development method***

FDD has five sequential activities during which the designing and building of the system takes place. The first three sequential activities establish an overall model shape. The

final two activities are iterates for each feature (Palmer & Felsing, 2002). A brief discussion of the five activities follows below.

- ***Development of an overall model***

The beginning of a project consists of an extensive walkthrough of the scope of the system and its context. Then follows detailed domain walkthroughs for each modelling area. Walkthrough models are then composed by small groups to support each domain. A single or combination of proposed models is selected, which becomes the model for that particular domain area. Domain area models are combined into an overall model.

- ***Build a Features List***

The information gathered from the overall model is used to identify a list of features. To achieve this, the domain has to be functionally broken down into respective subject areas. The subject areas each contain business-activities steps. Categorized feature list is formed from the steps contained within each business activity. The features are small pieces of client-valued functions represented in the form <action> <result> <object>, i.e. “generate unique number for an order", or “Calculate the total cost of an order”.

It should take at most 2 weeks to accomplish a feature Decomposition cannot exceed the upper limit of 2 weeks. If, during decomposition it appears that a feature may take longer than two weeks, then it must be decomposed further. Most features take much less than two weeks.

- ***Plan by feature***

This feature involves the creation of a high-level plan. The feature sets are ordered in accordance to their dependencies and priorities. In this phase, features or feature sets are assigned to chief programmers. The process where feature sets are ordered is what is referred as Class Ownership.

- ***Designing by feature***

The chief programmer selects a small group of features that should be developed within

a timeframe of about two weeks. The chief programmer and the class owners work out a detailed sequence diagram for each feature and modify the overall model if necessary. The next step is to write the class and method prologues and, finally inspect the design. The deliverable from the phase is a design package for each feature.

- ***Building by feature***

Actual coding of the classes by the class owners takes place during the building-by-feature phase. The output is a completed client-valued function unit which is then tested and its code is inspected. The completed feature is approved and is sent to the main build.

- **Systems development technique**

FDD uses regular UMLs with colour-encoded classes that are divided into different categories of unique colours. The use of colour with this technique allows quick understanding of the problem domain's dynamics. Interfaces and auxiliary classes are colourless (Palmer & Felsing, 2002).

2.2.4.2.1 Use of FDD

FDD is a methodology applicable to teams where the developers' experience varies, critical bigger projects, environments that demand waterfall development process, projects that are upgrading and enhancing existing source code, and projects that aim to develop the next version of an existing application (Abrahamsson et al., 2002). However, Abrahamsson et al. (2002) also note that very few reports exist of organisations that have successfully implemented FDD.

2.2.4.2.2 Strengths and weaknesses of FDD

FDD is designed to enable teams to deliver frequent (every two weeks) and tangible results without compromising quality. FDD is an extremely iterative process that is results oriented and focuses more on people than large documentation. FDD discards the traditional approaches of separating the domain and business analysts from designers and implementers. Instead, analysts are removed from their abstractions and

placed in the same room as implementers and users.

To date, research has uncovered limited weaknesses in FDD. However, there are some identified limitations to the methodology which can be attributed to its relativity of use. FDD is designed to accommodate much larger team sizes. One limiting factor could be a high reliance on the Chief Programmer, as they perform other roles of coordinator, mentor and lead practice, FDD teams scale well to much larger project teams, therefore for a small team it would not be as affective (Palmer & Felsing, 2002).

2.2.4.3 Dynamic Systems Development Methodology

DSDM was developed in the United Kingdom in 1994. It is a non- proprietary and non-profit framework for Rapid Applications Development (RAD), maintained by the DSDM Consortium.

- ***Systems development approach***

DSDM was derived from rapid application development practices and it is also an extension of these practices. DSDM is seen in the light of a framework, more than a method. DSDM addresses common issues commonly faced by agile development methodologies. It firstly states explicitly the difference between DSDM and traditional methods with respect to flexible requirements. The main idea behind DSDM is to fix time (time boxes) and resources to a product and then adjust the functionality accordingly (Abrahamsson et al., 2002).

- ***Systems development process model***

DSDM is an iterative and incremental approach that embraces principles of agile development, including continuous user/customer involvement.

- ***Systems development method***

There are seven phases in which a project goes through in its DSDM life cycle. The phases include: Pre-project, Feasibility study, Business study, Functional model iteration, Implementation, Design and build iterations, and Post project (Cohen *et at*., 2004).

- ***Pre-project***

In this phase, the project readiness is established. This includes determining the availability of funds to tackle the magnitude of the proposed project, and whether all other necessary factors are in place to begin a project.

- ***Feasibility study***

In DSDM, the feasibility study should not exceed a few weeks. During this phase, various factors are considered in deciding whether DSDM is the correct approach for the specific project (Stapleton, 1997).

- ***Business study***

The business study phase is intensely collaborative and uses a series of workshops, attended by knowledgeable staff as well as the people at the head of the organization, to reach consensus on what the priorities are of the development. The result during this phase is the Business Area Definition, identifying the users, markets, and business processes affected by the new system.

- ***Functional model iteration***

This phase builds on the high-level requirements identified in the business study. The DSDM framework functions by building a number of prototypes based on risk. These prototypes are then evolved into the complete system. This phase and the design-and-build phase shares a common process:

1. Identify what is to be produced
2. Agree on when and how to do it
3. Create the product
4. Verify whether it has been produced correctly. This is done by reviewing documents, demonstrating a prototype, or testing part of the system. (Cohen *et al.*, 2004)

- ***Implementation***

The DSDM framework functions by building the prototypes into final products. As the prototypes are being implemented, there is constant reviewing of the business needs, checking with the users for their approval. Once the prototypes have evolved into the complete system, the users need to be trained.

- ***Post-project***

Post-project phase ensures that the system developed is working efficiently and effectively. To achieve this, continuous maintenance, enhancements and fixes in accordance with DSDM principles must be applied.

Figure 2.3 below shows the phases that a project goes through as discussed above.

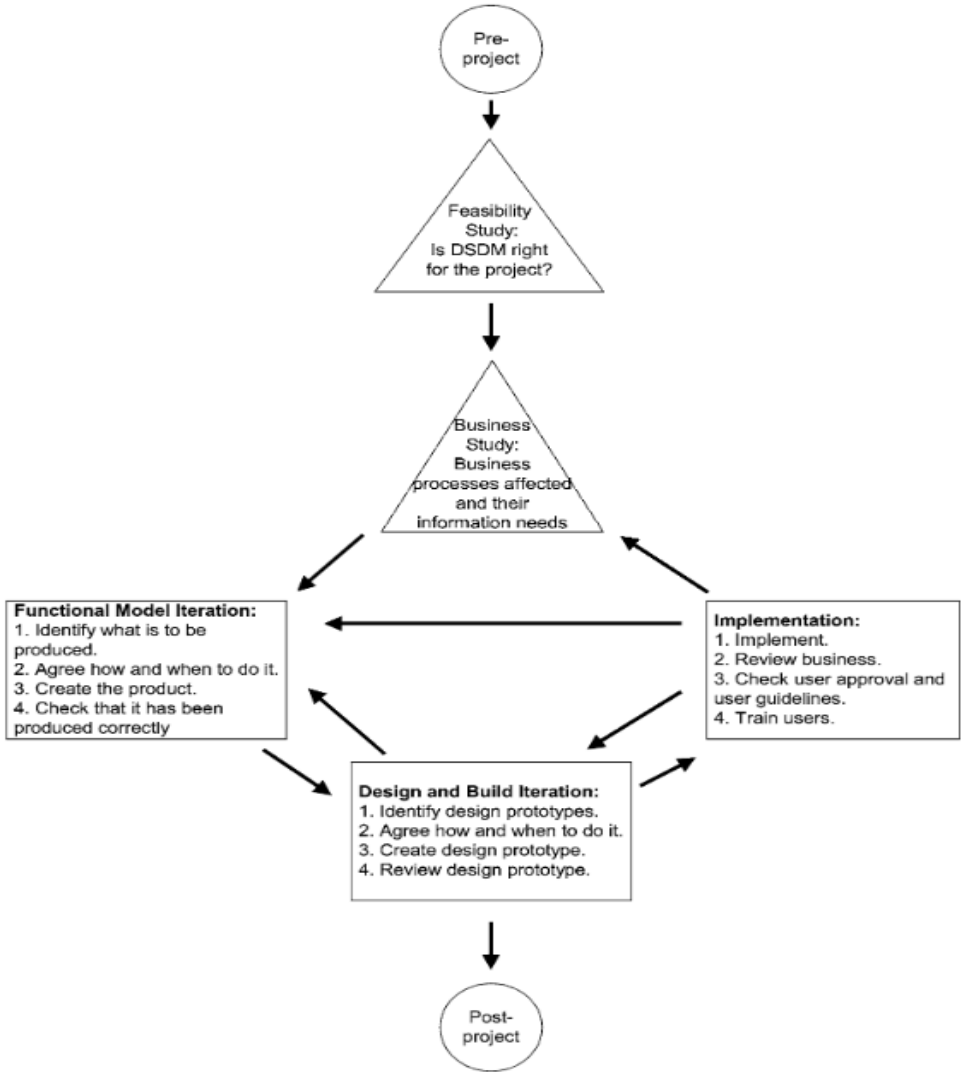

Figure 2.3: The DSDM Lifecycle (Cohen, et al., 2004)

- ***Systems development technique***

Prototyping is a key technique in DSDM as it produces a model of how the proposed system will function. Users find this extremely useful in visualizing the new system. Prototyping is very strong in areas where other methods are weak.

2.2.4.3.1 Uses of DSDM

According to Abrahamsson et al. (2002), DSDM is better suited to development of small or large business systems rather than to scientific or engineering projects. Large projects can be split into smaller units and assigned to smaller teams. Team sizes may

vary from a minimum of two and up to six members, and there may be many teams involved in a project. Each team ought to have at-least one user and one developer.

2.2.4.3.2 Strengths and Weaknesses of DSDM

Most of the method's strengths have already been discussed. I in summary; DSDM is advantageous because it is an iterative–incremental process with active user involvement. In DSDM, prototypes are used to collect information rather than lengthy documents (Highsmith & Cockburn, 2001). It is suitable for projects with highly volatile requirements, since it is easily adaptable.

However, DSDM is not scalable and has inflexible constraints on resources and time. Prototyping takes precedence over other models such as visual models (Stapleton, 2003).

2.2.4.4 Lean Software Development (LSD)

Lean development was founded by Robert Charette (Cohen *et al.*, 2004); it applies lean development principles of the Toyota Product Development System to software development (Poppendieck, 2007). According to Windholtz (2005), a revolutionary new concept was introduced in the 1950s by the Japanese; the Japanese business Toyota® Motor Company discarded the old way of testing (that is, only at the end of a production cycle) and replaced it by testing at every phase in the cycle. Batch sizes were reduced to reduce cycle times, which allowed for defects to appear sooner, and could thus be corrected faster. This key change reduced the number of components assembled with the same defect. This means less rework and lower costs.

Still today, most software is built in long cycles ranging from months to years - and often testing is left until the final step. Lean Software Development provides a different vision, states Windholtz (2005). This vision includes quick cycles; rapid, continuous, automated testing; and, close interaction with the customer. This provides the software that the business needs at the time when it is needed.

The following sections are guided by the work of Poppendieck and Poppendieck (2006); they apply Lean Manufacturing principles to the Lean Software Development environment. Lean Software Development Methodology does not follow actual

processes; however, it is built around seven guiding principles which will be discussed further in the next section.

- ***Systems development approach***

LSD does not really follow a development approach; however, there are underlying principles on which the methodology is built around. Poppendieck and Poppendieck. (2006) identified seven principles namely: Eliminate waste, Build quality in, Create knowledge, Defer commitment, Deliver fast, Respect people and, Optimize the whole.

- ***Eliminate Waste***

Kumar (2005) states that the first step in lean thinking is to understand the concept of "value", and what activities and resources are necessary to create it. Since no employee would like their tasks to be considered a waste, the task of determining what "value" is and what adds value, needs to be done at a high level by senior management. The manufacturing industry provides seven examples of waste, namely: overproduction, inventory, extra processing steps, motion, defects, waiting periods and transportation.

In software development, the criteria for what constitutes waste are similar to the seven wastes identified in the manufacturing industry. The wastes arising in software development should be avoided by project teams at all costs. Wastes in the manufacturing industry can be adapted to represent wastes of the software-development process: overproduction may equate to development of extra unnecessary features; inventory represents the requirements; extra processing steps is similar to performing extra steps in manufacturing; motion can be seen as the process of finding new information; defects are bugs undetected during testing; waiting for management and customer decisions is time wastage; and finally, transportation may refer to handoffs.

- ***Build Quality In***

This principle focuses on building quality into the software code from the beginning, rather than testing for quality at a later stage (Poppendieck & Poppendieck, 2006). The development team should not focus on minimizing defects in a system; but rather to avoid the creation of defects in the first place. Defects need to be caught as early as possible and this may lead to halting of production if needs be, to achieve this goal.

- ***Create Knowledge***

Software development is a continuous learning process. The best way of improving a software development environment is to encourage learning. The learning process is speeded up by making use of short iteration cycles. Each cycle should be coupled with refactoring and integration testing. Feedback can be improved by short feedback sessions with customers, when determining the current phase of development and future improvements.

- ***Defer Commitment***

This principle can be seen as a Just-in-Time decision principle. According to Poppendieck & Poppendieck (2006), important decisions should be made only at the last possible moment. By delaying the time it takes to make the decision, you create more time to gather the necessary information required to make the best decision and choose the best option. However, not all decisions should be deferred, but an attempt should be made to have decisions reversible. This way, rolling back a few steps would allow the right decisions to be made.

- ***Deliver Fast***

The sooner a defect-free product is delivered, the sooner feedback can be gathered from the customer so that any necessary improvements can be made or new requirements can be added to meet the customer's needs. Shorter iterations improve learning and communication.

- ***Respect People***

This principle is very simple to implement if you work in a team where every vote counts and no one person is regarded as more important that the next. For instance, Mr X's proposal cannot be deemed weaker than Mr Y's just because Mr Y has worked longer at a company than Mr X. Everyone's opinion must be given a fair chance to be heard, otherwise you might lose out on a brilliant plan or strategy.

- ***Optimize the Whole***

A lean organization optimizes the whole value stream, from the time it receives a requirement from a customer until the software is deployed and the requirement met. The overall value stream would be affected if an organization is to focus on optimizing less than the entire value stream.

- ***Systems development process model***

An iterative development process model is followed in LSD. This model ensures development of an overall system of the highest quality and at the fastest rate possible.

- ***Systems development method***

There does not exist any established practices or guidelines that need to be followed in LSD. The aim of LSD is predominantly on promoting principles of the methodology. Having no set development lifecycle, LSD can thus be adopted by an existing methodology.

- ***Systems development tools and techniques***

Lean Software Development does not prescribe tools or techniques. Thus lean development cannot be considered to be a software-engineering methodology per se, but rather a fusion of a system of practices, principles, and philosophy for building software systems for a customer's use (Poppendieck & Cusumano, 2012). Lean developers need to think outside the box, thus allow flexibility.

2.2.4.4.1 Use of LSD

This SDM is a set of principles that can be applied to almost any kind of system that needs to be developed by a business for a client in the least amount of time and at the lowest cost. The methodology is useful for amplifying learning, when decisions can be made as late as possible, for empowering teams, building integrity, and when there is a need to see the whole product (Dybå & Dingsøyr, 2008).

2.2.4.4.2 Strengths and Weaknesses of LSD

The greatest strength of this ASDM is that lesser defects are produced in the code while costs are cut to boost total profit.

The drawback to this approach is that the whole production line should be rebuilt and set up according to the principles, to achieve the goals set by this ASDM. But if this drawback is met head on, you will have a permanent advantage as the transition needs to be done only once and then just maintained.

2.2.4.5 Scrum

- ***Systems development approach***

The first reference to Scrum methodology emerged from an article published by Takeuchi and Nonaka (1986). The publication revealed a new, adaptive, quick, self-organizing product development process methodology called Sashimi (after the Japanese way of presenting sliced raw fish, where each slice overlaps on the slice before it)

Sashimi originated in Japan where the Waterfall (Traditional) model was inadequate for the product-development process. The number of phases was reduced to four - requirements, design, prototype, and acceptance - without removing any activities, which resulted in an overlap of the Waterfall phases. However, some literature links the name of the methodology with the sport of rugby; a “scrum” is a strategy for reinstating a dead ball back into play using teamwork,’ and this is believed to be the origins of the methodology “Scrum” (Awad, 2005).

Scrum is a management framework for incremental product development which can accommodate robust teams of about seven people (James, 2009).

- ***Systems development process model***

Scrum is an incremental and iterative process for systems development. Scrum’s focus is on how the team members are to function in order to produce a flexible system in an environment where requirements are constantly evolving (Awad, 2005).

James (2009) points out that Scrum teams make use of fixed-length iterations called Sprints. Sprints are approximately between 14 to 30 days long. Each Sprint goes

though the traditional phases found in the Waterfall methodology. The expected output from a sprint is a potentially shippable, tested product increment at each iteration phase. The number of Sprints required to reach the end goal can vary from 3 to more than eight Sprints.

- ***Systems development method***

According to Abrahamson *et al.* (2002), there are three phases that a project goes through. These phases are: pre-game, development and post-game.

Schwaber and Beedle (2002) describe each phase found in the Scrum process:

- Pre-game phase, which itself is divided into two inner phases:
  - Planning phase: This phase gives a description of the system to be developed and establishment of the Product Backlog List. The list of requirements is prioritized and the amount of work needed to implement them is estimated. This phase also describes the project team, tools required, training requirements, risk assessment and controlling issues, and approval of verification management.
  - Architecture level design phase: This phase contains the high-level design and the architectural planning based on the Product Backlog Items. A design review meeting is held during this phase to cover the proposals for the implementation.
- Development phase: This phase is where the “agility” aspect of the whole process kicks in. Sprints take place in this phase. The different environmental and technical variables identified in Scrum are managed by means of various Scrum techniques during the Sprints. Note that there may be more than one team involved in building the next increment.
- Post-game phase: This phase is initiated once the Product Backlog List is empty and no additions of new requirements can be made to the list. At this point the system is ready to be released. Planning for the release should have been done

during the post-game phase.

- ***Systems development technique***

To complete a Sprint activity in Scrum, the scrum team applies some tools and techniques to aid the process. At the beginning of each sprint cycle the team uses Sprint Planning Meetings, Daily Scrum Meetings and Sprint Backlogs. In planning meetings, work to be done during the cycle is determined; Sprint Backlog stipulates a list of features assigned to a Sprint. The Daily Scrum meetings last about 15 minutes, to track progress of the scrum team and address any challenges (Schwaber & Beedle, 2002).

2.2.4.5.1 Use of Scrum

According to James (2009), “Scrum has been used for a variety of products, but has initially been most popular for software products using object-oriented technologies”. Further he states that Scrum is exceptionally suited for high-risk projects where traditional efficiency concerns are secondary to the ability to deliver the right product by the set deadlines.

Abrahamson *et al.* (2002) stated that the Scrum method is suited for teams of fewer than ten people - and if more people are available there should be more than one team.

2.2.4.5.2 Strengths and Weaknesses of Scrum

According to Schwaber and Beedle (2002), Scrum can be adopted to manage any engineering practices in an organization, a positive for Scrum methodology. However, they also note that, when implemented, Scrum changes the job descriptions of the people involved, and may result in confusion earlier on as to who is responsible for what tasks.

Another major strength of Scrum is that it works extremely well on new projects and existing projects (Schwaber & Beedle, 2002). The methodology may limit the size of the project due to the fact that every individual on the team needs to understand all of the steps involved in development.

The previous section described the properties of five selected agile system

development methodologies. Each methodology is defined based on the four components upon which an SDM is defined. The next section briefly compares agile methodologies to traditional system-development methodologies.

2.2.5 Comparison of agile and traditional methodologies

In many organizations, software development is evolutionary development of the software product within the organization, often within a particular technical design and architecture. However, the requirements and their prioritization are usually plenty, and this is problematic in evolutionary development. Therefore, in such conditions, agile methods tend to outweigh traditional methods in terms of speed and efficiency of development as well as quality of the product (Iivari & Iivari, 2010).

Comparing agile and traditional methodologies can best be done when the comparison is based on some sort of a theoretical framework. In this study, the comparison is structured when based on the theoretical definition of systems-development methodologies.

As stated in 2.2.1 (definition of an SDM), Iivari and Huisman (2005) defined a system-development methodology as a combination of approaches, methods, process models, and techniques. Each of these elements has already been discussed in section 2.2.1. Based on elements that define a systems development methodology, comparisons of agile and traditional systems development methodologies can be derived.

2.2.5.1 Comparison based on systems development approaches

- **Traditional Approaches**

Traditional software development approach is a process-centred approach that is guided by the beliefs that the origin of any variation can be identified and may be removed or reduced by constantly evaluating and refining processes (Cockburn and Highsmith, 2001).

Traditional methodologies follow structured and object-oriented approaches to system development. They are plan-driven software development methodologies not flexible to dynamically adjust the development process (Nerur *et al.,* 2005).

- **Agile Approaches**

Agile systems development approaches are people-centric and tend to favour an object-oriented approach. Unlike the traditional methodologies, Agile approaches are based on agreed guiding principles stipulated in the agile manifesto (BECK et al., 2001).

These principles are based on four values that compose agile methods:

- The supremacy of individuals and interactions over processes and tools.
- Emphasis on customer collaboration over contract negotiation.
- Preference of working software over comprehensive documentation.
- Responding to change over plan-driven follow-throughs.

2.2.5.2 Comparison based on systems development process model

In a nutshell, traditional methodologies spell out predictability, whereas agile methodologies spell out adaptability.

- **Traditional Process model**

The traditional systems development approach is steered by a life-cycle model such as the waterfall model and the spiral model.

The classic example of a traditional methodology such as the waterfall model follows a linear and sequential approach to software design and systems development. Each phase is allocated to a designated individual/ team to ensure monitoring and controlling of projects and deadlines within the project phases. This approach ensures that projects or deliverables are submitted on time.

- A linear approach means a phase-by-phase approach for product building. The phases through which systems go through in a waterfall model are:
- Analysis Phase: The project team analyses the business requirements/needs.
- Design Phase: Business requirements are translated into IT solutions and choice of technology is made.
- Implementation Phase: Coding takes place.
- Testing Phase: The code developed is evaluated by the end-user
- Evaluation and maintenance Phase: Ensuring that the code works properly.

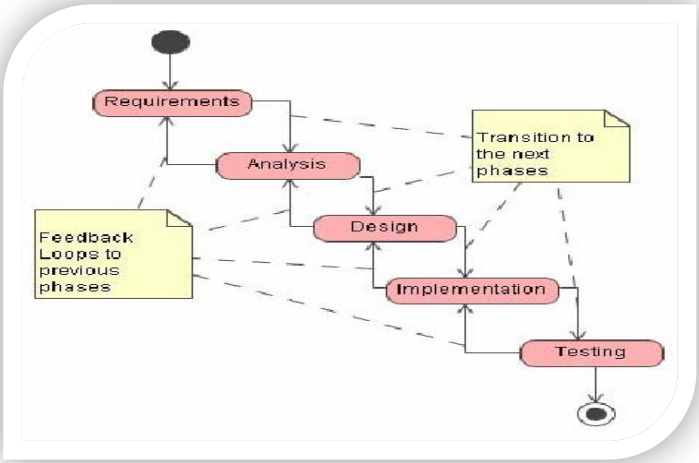

Figure 2.4: Classic phases in systems development methodology (Extracted from Klopper et al., 2007)

- **Agile Process model**

In traditional methodologies, instances where changes have to be made to the code or be the design phase would result in starting the process from the beginning. As impractical as it sounds, it is what would happen in a traditional methodology.

However, agile methodologies are low overhead methods that put emphasis on values and principles rather than processes. Agile methods are renowned for minimizing overheads such as documentation and meetings. Agile methods are suited to small teams with regularly changing requirements, as opposed to larger projects.

In agile methodologies, the process model involves teams working in short iterative cycles of development (weekly, monthly etc.), driven by product features. The project priorities are re-evaluated at the end of each cycle. The team meets to reflect and engage in collaborative decision making, feedback is incorporated, and changes are made to the system (Nerur *et al.,* 2005).

2.2.5.3 Comparison based on systems development methods

- **Traditional Methods**

Traditional methods are process-centred approaches that make use of guidelines, tools and techniques to accomplish tasks based on an underlying philosophy. Below are

some commonly-used methods:

- The Waterfall Model is a linear framework type that permits system development in sequential phases. The preceding phase must be completed before the next phase can begin.
- The Spiral Model is a combination of linear and iterative framework types. It is a heavyweight software development model that combines elements of design and prototyping-in-stages (Awad, 2005).
- Unified Process involves arranging all activities, including modelling, into workflows. The method allows development in an iterative and incremental manner (Awad, 2005).
- Prototyping is an iterative framework type. However, it cannot operate as a stand-alone method of developing systems. Prototyping is infused within other traditional methods such as in the spiral model.
- The Incremental approach is a combination of linear and iterative framework type. Initial phases are defined by the waterfall approach, then the rest of development phases are defined by prototyping.

- **Agile Methods**

Several methods have become prominent. Often, these methods are referred to as lightweight methods and try to overcome limitations encountered with traditional plan-driven approaches or the heavyweight methods.

Several agile software development methods exist. Some of the commonly known methods include:

- Rapid application development (RAD) is an iterative development method that makes use of prototypes. It is used in environments where requirements are rapidly changing.

- XP development process is characterized by short development cycles, incremental planning, constant user feedback, reliance on communication, and evolutionary design (Awad, 2005).

- Scrum is an iterative and incremental process for systems development. It focuses on how the team members are to function in order to produce the system flexibility in

an environment where requirements are constantly evolving (Awad, 2005).

- Dynamic systems development is a combination of RAD and extension to iterative development approaches. The core philosophy behind the method is to fix time and resources, and then adjust the functionality levels based on the time allocated and available resources.

- The feature driven development (FDD) approach is not involved in the entire software development process. Rather, it focuses on the design and building phases (Awad, 2005).

2.2.5.4 Comparison based on systems development techniques

- ***Traditional Techniques***

Traditional methodologies used object-oriented technologies on projects. Project management tools such as PERT diagrams are utilized to estimate schedules. Code editors and compilers are used for manipulating code.

Models such the “Unified Process” model makes use of visual modelling software, such as Unified Modelling Language (UML). Requirements are managed by making use of use-cases and scenarios. Other models such as the Spiral model make use of prototyping techniques.

- ***Agile Techniques***

Agile methodologies can use any programming language. However, they also favour object-oriented programming. Extreme Programming for example, uses pair programming which allows two developers to work together on one computer, which increases the chances of finding bugs, and leads to a simpler design.

Table 2.1 below compares traditional and agile methodologies, based on factors such as fundamental assumptions, control, style of management, communications styles, models upon which the methodologies are built, and technology.

Table 2.1: Comparing agile to traditional methodologies (*Nerur et al.,* 2005)

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

2.2.6 Summary

The previous section introduced agile systems development methodologies, and discussed the agile manifesto and some commonly used methodologies. This forms one part of the intended research. The next section will discuss the second core concept of study, being software process improvement models.

2.3 SOFTWARE PROCESS IMPROVEMENT MODELS (SPIMs)

2.3.1 Introduction

This section discusses the second aspect of this research - software process improvement models (SPIMs), also known as maturity models.

Organisations (small or large) have to manage and improve their software development processes to meet the requirements of the customer on time, at a lower cost, without compromising quality. Wang and King (2000) define software process improvement as "*a systemic procedure for improving the performance of an existing process system by changing or updating the process*". Humphrey (1991) defines SPI as "the set of activities, methods, and practices that aid developers in the production and evolution of

software".

In a nutshell, maturity models are used to develop and refine an organization's software development process. Software developing organizations strive to deliver a product that helps their customers gain a competitive advantage (al-Tarawneh et al., 2011). The adoption of SPIMs enables organizations to improve the software product, quality and capability of processes (Ruiz *et al.,* 2011).

The quality of the software as a product is strongly influenced by the quality of the processes followed in developing and maintaining the software. Statistical control is one basic principle behind software process improvement. If a process is under statistical monitoring, better results can be achieved by simply improving the process (Humphrey, 1988).

The software developing industry has seen a number of frameworks evolve from the principle of software process improvement. These frameworks differ in different aspects, and are thus difficult to objectively compare due to their comprehensiveness. These differences also make it difficult for an organization to determine which framework is ideal to adopt, and often the decision is subjective rather than objective (Halvorsen & Conradi, 2001).

Some of the SPIMs found in the industry include models such as the Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMi) (Huang & Han, 2005), International Organization for Standardization (ISO) 9000-3 (Kehoe & Alka, 1996), and Software Process Improvement and Capability Determination (SPICE) (Emam et al., 1998).

This research focuses on organizations of all sizes. In most small and medium enterprises (SMEs), the most widely-used model is CMM. The most extensively used ISO models are the 15504 (SPICE) and the 9001 (Pino *et al.,* 2008). According to Staples et al. (2007), CMMi is considered to be a heavyweight approach, and small organizations find it difficult to adopt such models. However, CMMi is currently replacing CMM and thus it is inevitable that small organizations will adopt CMMi.

Therefore, based on this wide and extensive use of these models, the next section of this chapter investigates in detail the three models commonly adopted by organizations; CMMi, SPICE and ISO 9000-3.

2.3.2 Capability Maturity Model Integration (CMMi)

The Software Engineering Institute (SEI) released capability maturity model integration (CMMi) in 2001 to incorporate existing capability maturity models (Huang & Han, 2005).

In CMMi, process management is the fundamental theme. CMMi is a standardized model for assessing and improving the software and systems development process within an organization (Ehsan et al., 2010). CMMi can be used to guide process improvement across an organization's division, a specific project, or an entire organization. CMMi cannot be considered as a process or as an application but merely as a set of guidelines (framework) that, when applied, will aid in improving the general structuring and processes of an organization.

2.3.2.1 CMMi Architecture Overview

The CMMi framework is composed of components that are used to build CMMi models, training materials and appraisal materials. The model components consist of core process areas, practices, goals and other useful information about the use of the model and its associated components. The training component consists of guidebooks and other supporting materials that aid in implementing the model. The appraisal component describes the process of appraising processes of the organisation against the goals and practices as described in the model component (CMMi Product Team, 2007).

The components that determine the CMMi structure for both staged and continuous representation are: process areas, specific practices, generic practices, specific goals, typical work products, sub-practices, discipline amplifications, notes, generic practice elaborations, and references (Alegrıa & Bastarrica, 2006). Figure 2.5 presents the CMMi model components.

The process area component is the main model component consisting of a group of related practices. These practices satisfy a set of goals for improving the area.

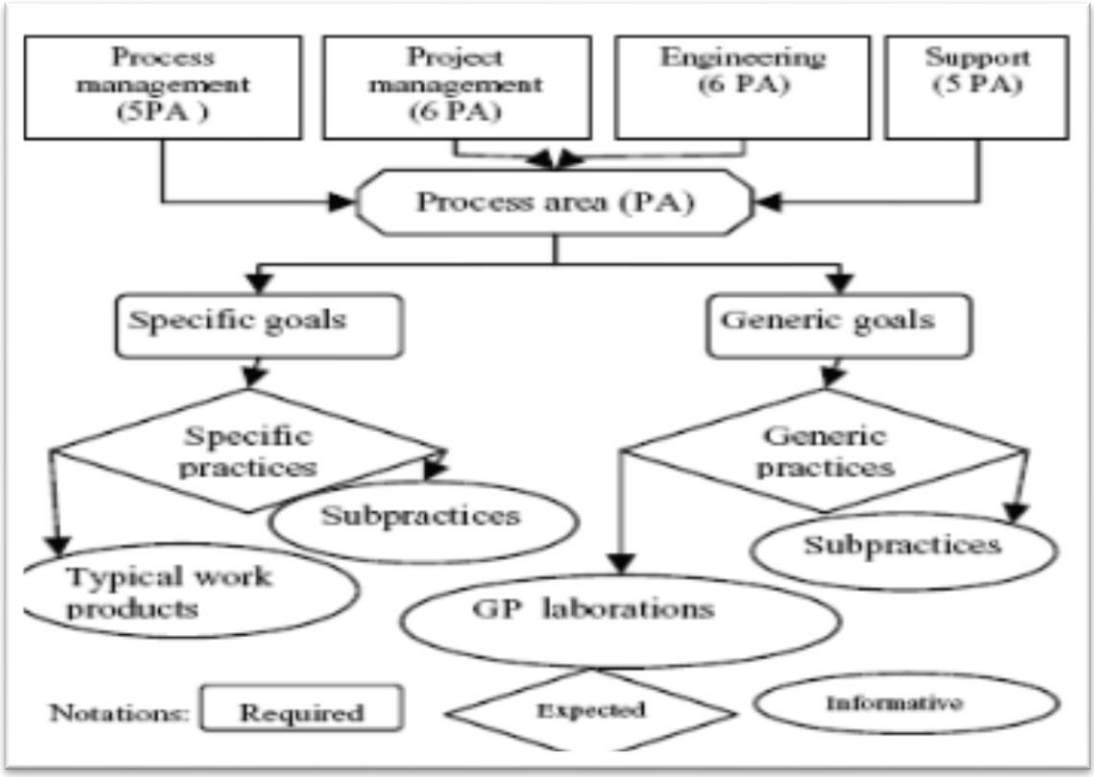

Figure 2.5: CMMi Components (Ehsan, et al., 2010).

There are three kinds of model components defined in a process area: Informative, Required and Expected.

- Required model components are requirements the organisation should achieve in order to satisfy that process area. Specific and General goals fall under the required component category.
- Expected model components outline what an organisation ought to implement in order to achieve/ satisfy its required components. They include general and specific practices.
- The Informative model components provide information that assists organisations in planning how to reach required and informative model components, such as sub-practices, discipline amplifications, typical work products, generic practice elaborations, goal and practice titles and notes, and references.

2.3.2.2 CMMi: Maturity levels

CMMi consists of five maturity levels; Initial level, Managed level, Defined level,

Quantitatively-managed level and Optimizing level. Each level and associated processes are briefly discussed in this section. Figure 2.4 below shows the CMMi maturity levels, from the initial level up to the optimizing level.

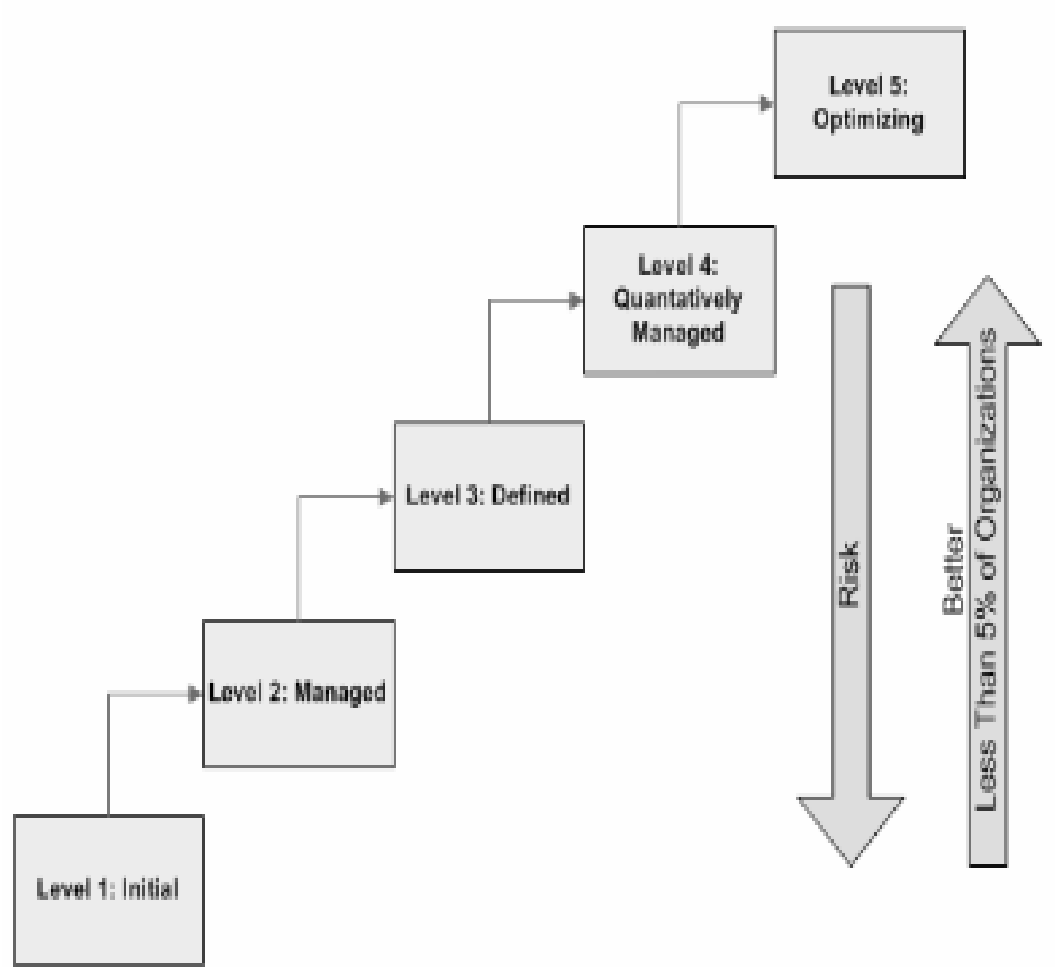

Figure 2.6: CMMi Maturity Levels (Extracted from Huang & Han, 2005).

CMMi model can be represented in two ways, namely Staged representation and Continuous representation (Huang & Han, 2005).

Staged representation of the CMMi model arranges process areas into five maturity levels which provide a recommended order for approaching process improvement:

1. ***Initial***: focus is on ad hoc processes and is not associated to any processes area.
2. ***Managed***: focuses on basic project management and is associated with Requirements management, Project planning, Project monitoring and control, Supplier agreement management, Measurement and analysis, Process and product quality assurance, and Configuration management process areas.

3. ***Defined***: focuses on process standardization throughout an organization. The process areas associated with this level are Requirements development, Technical solution, Product integrated, Verification, Validation, Organizational process focus, Organizational process definition, Organizational training, Integrated project management, Risk management and Decision analysis and resolution.
4. ***Quantitatively Managed***: focuses on Quantitative management and Organization process performance, and Quantitative project management are the process areas arranged at this level
5. ***Optimizing***: focus is on continuous process improvement: Organization innovation and deployment, and Causal analysis and resolution are the process areas concerned at this level.

Continuous representation has the same process areas as in the staged representation. No process areas are assigned to a specific maturity level; instead, the process areas are arranged into four process-areas categories, namely project management, engineering, process management, and support. Figure 2.5 presents the categories with the associated process areas and the maturity level at which each is expected to be performed.

Category	Process area	Maturity level
Project management	Project planning (PP)	ML 2
	Project monitoring and control (PMC)	ML 2
	Supplier agreement management (SAM)	ML 2
	Integrated project management (IPM)	ML 3
	Risk management (RSKM)	ML 3
	Quantitative project management (QPM)	ML 4
Process management	Organizational process focus (OPF)	ML 3
	Organizational process definition (OPD)	ML 3
	Organizational training (OT)	ML 3
	Organization process performance (OPP)	ML 4
	Organization innovation and deployment (OID)	ML 5
Engineering	Requirements management (REQM)	ML 2
	Requirements development (RD)	ML 3
	Technical solution (TS)	ML 3
	Product integrated (PI)	ML 3
	Verification (VER)	ML 3
	Validation (VAL)	ML 3
Support	Configuration management (CM)	ML 2
	Process and product quality assurance (PPQA)	ML 2
	Measurement and analysis (MA)	ML 2
	Decision analysis and resolution (DAR)	ML 3
	Causal analysis and resolution (CAR)	ML 5

Figure 2.7: Process areas in CMMi continuous representation (Huang & Han, 2005).

Organizations have the flexibility to choose process areas to improve in the Continuous representation. Specific process-area achievement is measured by using six capability levels, numbered 0 through to 5: incomplete, performed, managed, defined, quantitatively managed, and optimizing (CMMi Product Team, 2002). These capability levels are briefly described below:

- **Level 0** - Incomplete: A process that is not performed or partially performed.

- **Level 1** - Performed: Process that is regarded as performed if it satisfies the specific goals of a process area.
- **Level 2** - Managed: Refers to processes which are performed, planned, managed, monitored, and controlled for specific projects to achieve specific goals.
- **Level 3** - Defined: This is a managed process tailored from the organisation's set of standard processes, based on the tailoring guidelines of the organisation.
- **Level 4 -** Quantitatively managed: A defined process that is managed and controlled using statistical and quantitative techniques and methods.
- **Level 5** - Optimizing: This is a quantitatively managed and improved process adapted to meet relevant current and projected business objectives. Focuses on continually improving performance of the process, using incremental and innovative methods.

2.3.2.3 CMMi Criticism

Despite its successes, CMMi too is prone to criticism. Early research indicated that a better software process is achieved when CMMi is implemented; however, it comes at a cost. CMMi is not a fit-all standardized solution, as its application depends on the software development process being followed and the type software (adapted or coded from scratch). The SEI in 2005 recommended that the implementation of CMMi ought to be accompanied with an appropriate cultural shift. This meant that organisations would have to change their ways of producing software, which initially would have a negative impact, as people and organisations are often resistant to change (Gefen *et al*. 2006).

Critics have criticised numerous features of CMMi. Two prominent ones suggest that: (1) CMMi exclusively focuses on the process and, (2) prefers large and bureaucratic firms. The first criticism of CMMi is that it tends to promote the process as the only factor in software development, thus excluding technology and people. This results in non-guarantee of software project success. Secondly, it can be said that CMMi is more suitable for large organisations. This criticism stems from the fact that the SEI was sponsored by the U.S. defence department - a government body that is large and often bureaucratic, and focused on promoting form over substance. Such sentiments also apply towards large enterprises, such as multinational corporations or monopolies.

CMMi is better equipped for measuring the capability of an organisation to fulfil software specifications. Bureaucracy can also be promoted to such an extent that anyone within the organisation may think they can do a specific task (managers writing code, for instance) by simply following CMMi, much as following a recipe. This may result in suppression of creativity in the development process, and consequently project delays (Höggerl & Sehorz, 2006).

2.3.3 Software Process Improvement and Capability Determination (SPICE)

Software Process Improvement and Capability Determination (SPICE), also known as ISO/IEC 15504, is a framework for assessing software processes. This framework was developed by the International Organization for Standardization (ISO) and the International Electro-technical Commission (IEC) (Huang & Han, 2005).

ISO/IEC 15504 is arranged around a reference model in a two-dimensional architecture comprising of a process dimension and a capability dimension (El Emam & Jung, 2001).

The process dimension is divided into five categories: engineering, supporting, customer-supplier, organization and management. A process can be described as a set of activities acting as base practices that contribute to achieve the intended goals of process (Simon, 1996). The processes are also defined with capability levels ranging from level 0 to level 5. The process capabilities levels also have process attributes, with the exception of level 0 - the Incomplete process which has no attributes.

2.3.3.1 Process Dimension

The process dimension of the model consists of 35 processes that are grouped into five process categories (Simon, 1996):

- Engineering process category: This category consists of processes which directly specify, implement and maintain a system and software product.
- Supporting process category: Consist of processes that enable and assist the performance of other processes on the project.
- Customer-supplier process category: Contains processes that directly impact the end user/ customer.

- Project process category: Consists of processes which initiate, co-ordinate and manage the project's resources.

Organisation process: These are processes which establish an organisation's business goals. These processes also develop process, product and resource assets which aid the organisation in achieving its business targets.

2.3.3.2 Capability dimension

The capability dimension of the model consists of six levels composing a scale of requirements, from the lowest Level 0 to the highest Level 5. Table 2.2 below contains the capability levels with their associated process attributes.

Table 2.2: SPICE capability levels with their associated process attributes.

Level ID	Title	Process Attribute
Level 0	*Incomplete process:* failure to achieve purpose of the process. No tangible outputs arise from the process.	None
Level 1	*Performed process*: Base practices considered to be performed. The purpose of the process is achieved but not rigorously planned nor tracked. Performance is dependent on individual knowledge and effort.	Process performance
Level 2	*Managed process:* process produces quality work products within defined timescales. Performance is planned and tracked; products conform to specified standards and requirements.	Performance management Work product management
Level 3	*Established process:* process performed and managed via a process defined by good software engineering principals.	Process definition Process resource
Level 4	*Predictable process:* defined process is consistently performed in practice within defined control limits. Detailed measures of performance are collected and analysed leading to ***quantitative*** understanding of process capability and enhance ability to predict performance.	Process measurement Process control
Level 5	*Optimizing process:* process performance is optimized to meet current and future business needs. The process achieves repeatability in meeting defined business goals. Establishes quantitative process effectiveness and efficiency targets for performance. There is continuous process monitoring	Process change Continuous improvement

The combination of two dimensions of the model, SPICE is able to analyse any software process according to the Level requirements, in order to determine process capability. However, there is no specified formal order in which processes should be performed, nor what level should be attained by a specific process, to define good process quality.

2.3.3.3 SPICE Criticism

As with other software process improvement models, the SPICE models is also prone to criticism. The value of assessment, using a software process model for organizations

just starting up, can be a waste of money. Therefore, in an immature organization the results are likely to be meaningless (Fayad & Laitnen, 1997).

The results of process improvement are often not visible within a short period of time; it takes time to see the results. SPICE is not a free product and training opportunities are scarce. Therefore there are few SPICE experts in the market (Ehsan *et al*., 2010).

2.3.4 International Organization for Standardization (ISO) 9000-3

The ISO 9000 series of standards stipulate guidelines for quality product requirements that have to be adhered to when two parties (customer and supplier) come into contact with each other, such that the supplier demonstrates the capability to design and supply a quality product (McAdamand & Fulton, 2002). ISO 9000-3 provides guidelines for applying ISO 9001 to the development, supply, and maintenance of software. ISO 9001 is a standard found in the ISO 9000 series. ISO 9001 is a globally recognised standard for Quality Management Systems (QMS) which has been implemented by over a million organisations spanning 180 countries worldwide (Manders *et al*., 2015). Figure 2.8 provides the standards found within the ISO 9000 series.

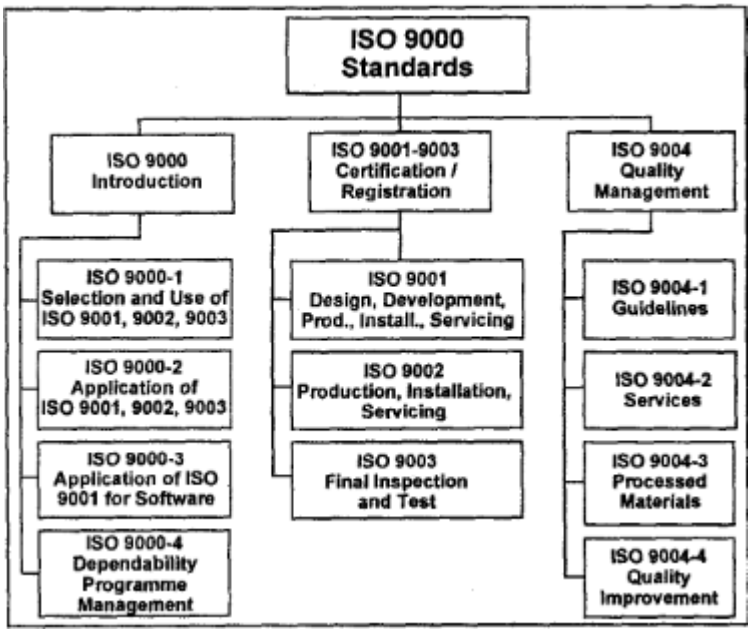

Figure 2.8: Overview of the ISO 9000 series of standards (Stelzer et al., 1996).

ISO 9000-3 is a set of prescribed standards on managing and assuring quality during

the process of developing, supplying, and supporting and maintaining of computer software, published by the International Organization for Standardization (ISO) in Geneva 1991(Nahler, 2009).

ISO 9001 comprises 20 clauses that provide the minimum requirements for developing a quality management system for use in software development, support and maintenance. When these minimum requirements are satisfied, the organisation may be classified as ISO 9001 certified. As discussed above, ISO 9000-3 contains software-specific guidelines for the use of ISO 9001(Ho-Won & Hunter, 2001).

ISO 9000-3 provides a model for bridging the gap between the ideal and real worlds. The roles and responsibilities of the supplier and the customer are laid out in the ISO 9000-3 guideline, which describes the software development process that can be tailored, implemented and customised in various ways (Kehoe & Jarvis, 1996).

ISO 9000-3 does not dictate to software manufacturers on how to analyse, design, code, test and document software products but rather emphasises that the manufacturer has a process that is clearly defined, documented and followed. A main concept of ISO 9000-3 guideline is that software development and maintenance is a diverse and integrated engineering process that is made up of unique phases, procedures, steps, and other activities.

2.3.4.1 ISO 9000-3 Guidelines/Clauses

The ISO 9000-3 guide contains software-specific requirements/processes that must be fulfilled for an organisation to be certified. Below is a brief discussion of ISO 9001 clauses of which ISO 9000-3 makes use (Paulk, 1995; Pfahl, 2008):

- Quality management system: In ISO 9001, the organisation is required to establish a well-documented quality system that includes a quality manual and plans, procedures and instruction. However, in ISO 9000-3, quality management is an integrated process throughout the project's life cycle.
- Management responsibility: Management is responsible for defining, documenting, understanding, implementation and maintenance of a quality policy. Management must also describe the responsibilities and authority for personnel tasked to manage, perform and verify work that affects quality as well as other aspects

concerning customers, and identify and provide verification resources.

- Software development and design: The organisation must develop and document procedures that will control development process and ensure that requirements are met. Planning procedures must also be designed and developed. They must identify teams that will be responsible for circulating, reviewing product design and development process. They must design procedures to control product outputs, plan for reviews and verification. They should have in place procedures that will validate the assumption that the product developed satisfies customer requirements.
- Product Inspection and testing: Organisations must continuously inspect and test products throughout their life cycle.
- Corrective and preventive action: Organisations must identify causes of non-conforming products, and implement corrective preventive measures. Corrective actions aim to eliminate actual non-conformities while preventive actions are to eliminate the causes of potential nonconformities.
- Training requirements: Personnel in an organisation may require training in order to perform certain tasks. Therefore, organisations need to identify training needs.
- Statistical Techniques/Measurement: Organisations must identify appropriate statistical techniques that can be used to verify the acceptability of process capability and product characteristics.

Table 2.7 below compares the extent to which the software development processes are covered between the ISO 9001 standards and the ISO 9000-3 guidelines.

Table 2.3: Software development processes covered in ISO 9001/9000-3 (Coallier, 1994)

Process	ISO 9001	ISO 9000-3
Measurement	Not covered	Guidelines on product and process measurement
Requirements management	Highly covered	Only focused on contractual approach
Project management	Highly covered	Basics of project management knowledge areas.
Subcontractor management	Highly covered	Basics guides to software development subcontracts
Quality system	Coverage mainly on conformance to specifications, requirements, and documented processes	Guidelines for software quality planning
Design	Non coverage	
Implementations	Not covered	Recommendation that adequate methods and too are be used
Verification and validation	Basics of design reviews	Basic guidelines on testing and acceptance testing
Configuration management	Basic document monitoring	Essentials of source-code configuration management
Maintenance and support	none	Basic guidelines on corrective and adaptive maintenance

2.3.4.2 ISO 9000-3 Guideline Criticism

Kohoe *et al* (1996) identify three criticisms of the ISO 9000-3 guideline: The guideline tends to misuse the term “quality” in the sense that any person, practices, activities and procedures involved in specification, development and maintenance of the software product is clearly engaged in activities that impact quality. However, since everything is associated to quality, there is a distortion between those engineering activities used to develop and maintain a product (e.g. analysis, design, code etc.) and those that assure

the quality of the product (and its development process such as audits, inspection etc.).

The detailed software-requirements phase is one often accepted step in software engineering which the ISO 9000-3 guideline overlooks. Thirdly, the guideline requires a quality plan; the quality plan identifies topics which belong in the documents that define engineering process or specific plans for developing and testing a product.

In summary, apart from critiques identified, Stelzer *et al.* (1996) also state that IS0 9000-3 is not the best solution for software development process. The quality managers only read its clauses once studying the IS0 9000 series of standards. When they realize that IS0 9000-3 is difficult to read in comparison to other parts of IS0 9000, quality managers tend to disregard ISO 9000-3 and opt for IS0 900 1. These standards are always undergoing refinements and ISO 9000-3 has become obsolete (and as from 2014 was renamed ISO 90003). The next section briefly compares the SPI models.

2.3.5 Comparison of Software Process Improvement Frameworks

The comparison of SPI frameworks generally attempts to distinguish the similarities and differences that exist between the models. However, the major differences lie in the level of detail and point of view (Halvorsen & Conradi, 2001).

According to Halvorsen and Conradi (2001), they identified four classes from which frameworks can be compared:

- Characteristics comparison method involves comparing attributes of the frameworks. The characteristics should preferably be comparable, measurable and objective.
- Bilateral comparison: in bilateral comparison, two frameworks are compared to each textually. This is achieved by comparing textual phrases.
- Needs-mapping involves comparing the needs of the users (organizational and environmental needs) to the properties of the framework.
- Framework mapping: most structured frameworks consist of defined sets of statements. The process of framework mapping involves creating a map from statements or concepts of one framework to those of another. This process can be achieved in two distinct ways by:
    - Mapping existing frameworks into a base framework designed for comparison purposes
    - Mapping existing frameworks

Halvorsen and Conradi (2001), in their proposed taxonomy, identified 25 characteristics upon which frameworks can be described. These characteristics are further subdivided into categories:

- *General Category*
  This category describes general attributes or features of the SPI framework that are often specific to each framework. Such as geographical origins, scientific origin, adaptability etc.
- *Result Category*
  This category deals with the outcomes of employing SPI in an organization, such as; goals for employing that specific framework, process artefacts, certification, implementation costs and validation.
- *Process Category*
  Describes how the framework is used. The characteristics include assessment, assessor, Process Improvement Method, Improvement focus, Improvement initiation and Analysis Techniques.
- *Quality Category*
  Deals with quality dimensions by identifying aspects related to quality such as measuring progression, quality perspective and comparative.
- *Organization Category*
  The characteristics in this category are directly related to those attributes of the organization and its surrounding environment in which the SPI framework is used, such as who the stakeholders are, the size of the organization, and coherence. It asks the questions: What are you going to do with the above information? How are you going to apply it?

2.3.5.1 Comparing CMMi and ISO/IEC15504 SPICE

The Capability Maturity Model Integration and SPICE models can be mapped into one another. According to Ehsan *et al*. (2010), after conducting a direct comparison between the two models, they discovered that the two models mapped directly into one another. This comparison is presented in Table 2.4 below.

Table 2.4: Comparing CMMi and SPICE (Ehsan et al., 2010)

CMMi Continuous Representation	SPICE
Processes Categories	
Process Management	Organization
Project Management	Management
Engineering	Engineering
Support	Support
	Customer-Supplier
Capability Levels	
Optimizing	Optimizing
Quantitatively Managed	Predictable
Defined	Established
Managed	Managed
Performed	Performed
Incomplete	Incomplete

It is evident from the mapping that both models implement the same number of capability levels similarly named. The process category is also similar, with the exception of an additional process in SPICE. Research carried out by Ehsan *et al*. (2010) also revealed differences in the two models. For instance, some process areas covered in SPICE are not found in CMMi. These process areas include Operational Process, Management Process, and Process Alignment Process. The researchers also established that the following process areas are partially addressed in CMMi: Supply Process, Software Maintenance Process, Human Resource Management Process, and Reuse Process.

2.3.5.2 CMMi or SPICE

Apart from structural differences discussed above, general differences exist between CMMi and SPICE. Ehsan *et al*. (2010) conducted a software industry survey, with the aid of questionnaires, as data-generation methods to discover which of the two models was more popular in the industry. The results obtained showed that CMMi is more commonly used in the software industry, compared to SPICE, for the following reasons:

- CMMi is freely available.
- There are few SPICE-trained professionals in the market
- There is little industry awareness about SPICE
- Organizations must pay for SPICE
- Adequate Process Improvement was seen in the software houses that implemented CMMi.
- Trained resources and quality professionals of CMMi are available.
- Appraisal of CMMi is relatively easier to call upon, compared to SPICE.
- Management was satisfied with the outcome of CMMi and felt that customers feel better in dealing with a certified company.

In summary, it is evident that CMMi outweighs SPICE in terms of cost, availability, awareness and appraisal, thus making CMMi the preferred model. Similarly, the criticism of ISO 9000-3 by Stelzer et al. (1996) states that IS0 9000-3 is not the best solution for software developing houses.

2.3.6 Interrelations between ASDMs and SPIMs

To compare agile methods to SPIMs is not to “compare apples with apples”. Agile is a software development philosophy, whilst SPIMs are maturation models. The two concepts are incomparable to some degree as their focus is not the same. For instance, CMMi focuses on the development process of an organisation in its entirety, whilst agile methodologies can be a framework or a portion of a single development process. However, despite the different focuses, they have interrelations and can coexist and support each other (Therese & Alagarsamy, 2011). Leithiser and Hamilton (2008) argue that the two approaches should not be considered incompatible despite the differences, but instead acknowledge the fact that they have the potential to complement each other.

A combination between the two approaches can be beneficial to an organization, as it may provide a better software development process, improved software quality, better project management methods and ultimately reduced development costs (Koutsoumpos & Marinelarena, 2013). The following sub-sections describe some of these interrelations.

2.3.6.1 CMMi vs. Agile Methods

Most organisations face the pressure of conforming to increasingly difficult functional requirements while at the same time expected to retain acceptable quality. As much as CMMi has become an important model for organisations to be competitive, not all organisations need to achieve a certain level of compliance in order for them to compete effectively. As discussed earlier, agile methods are suited for situations where a rapid solution is needed, while CMMi is often used for longer and more complex projects that are rigid in nature and can afford the extra overheads (Leithiser & Hamilton, 2008).

Leithiser and Hamilton (2008) identified three CMMi and Agile “trade-offs questions” to be asked when deciding between the two:

- Will selection of one approach inhibit the use of the other?
- When is the project too large or critical for Agile methods, or too small for CMMi?
- What are the key decision factors?

CMMi is associated with five maturity levels discussed in section 2.3.2.2. For example, Level 2 in CMMi can be attained without major adaptations if on XP or Scrum. However, it is practically impossible to reach Level 4 and 5 with XP and Scrum without making changes to the methods that contradict agility (Fritzsche & Keil, 2007).

CMMi and Scrum can be implemented together by mapping Scrum practices into CMMi process areas. The results are better adaptability and predictability. However, for the combination to be successful, adaptations to the Scrum practices (relating to issues management, agile risk management and estimates methods) need to be done (Lina & Dan, 2012). The ideas of this interrelation stem from the fact that CMMi is beneficial when applied at organisational level to encompass all processes of development. While Scrum does not focus on the level of the organisation, its practices can be used to

improve CMMi's plan-driven processes. Scrum can identify potential risks but has no practices to mitigate the risks, whilst CMMi has strategies to combat and manage risks. The benefits gained are better ways of managing risks, issues and estimation management, better documented requirements, and a better-quality of product. Scrum brings about better communication with the customers (Marcal et al., 2007; Lina & Dan, 2012).

Alegrıa and Bastarrica (2006), in an attempt to establish the extent to which agile methods can contribute to CMMi implementation, took two different approaches - a specific and a general approach - on how agile process assets can fulfil CMMi requirements individually or combined. Figure 2.9 shows the relationship between CMMi requirements and agile process assets.

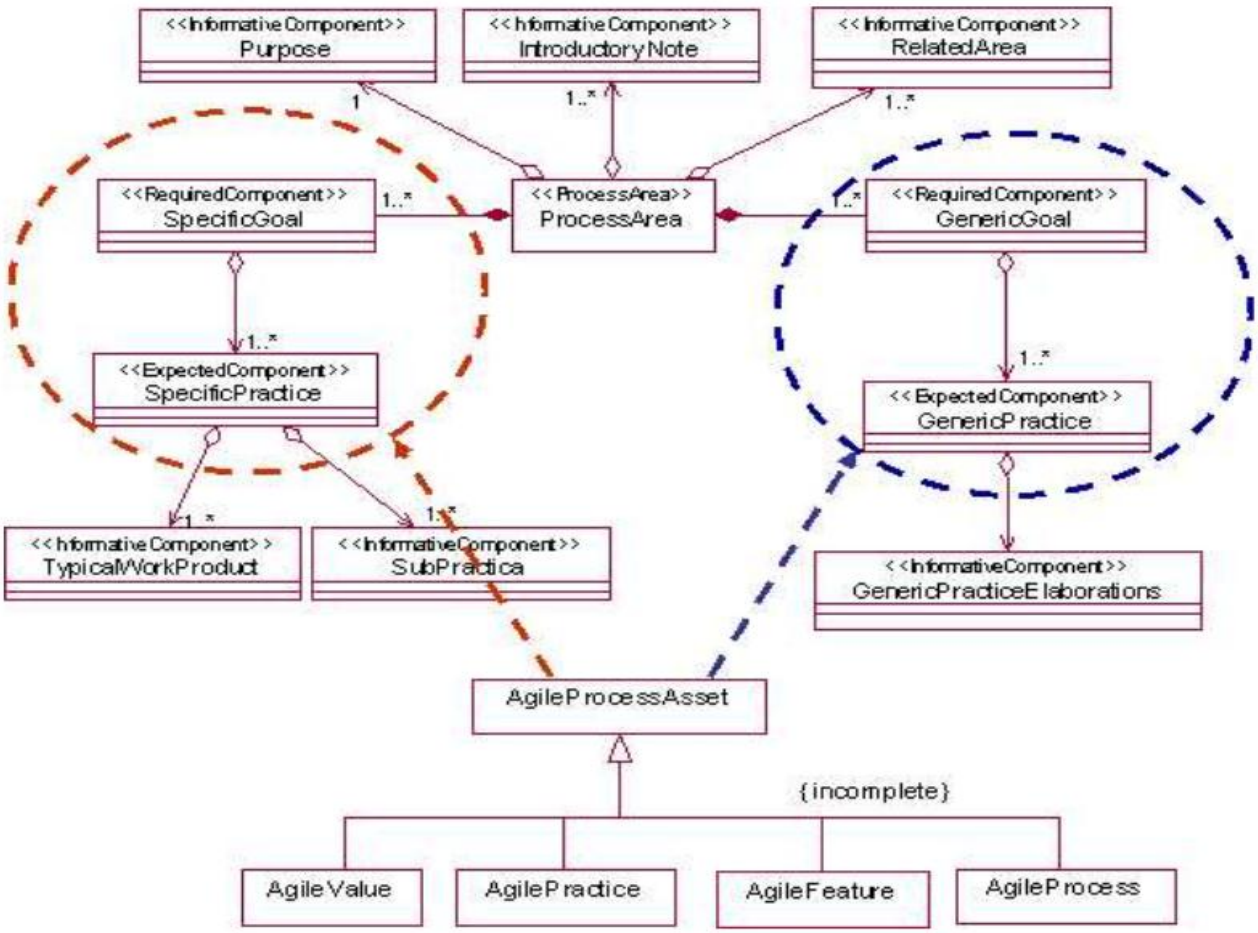

Figure 2.9: Relationship between CMMi requirements and agile process assets (Alegrıa and Bastarrica, 2006)

The specific approach (left red circle) looks at the CMMi model components that provided detailed information about the fulfilment of specific goals and practices. Meanwhile, the general approach (right blue circle in figure 2.9) looks at the general description of each process areas, their general goals and generic practices. This approach gives an idea of how agile assets could allow an organisation to reach a CMMi certification, at what level, and which requirements are not covered.

Their results gathered, using the general approach, shows that agile practices can implement the Requirements management process area. Project planning is covered by most of the ASDMs. Monitoring and control is not as organised with ASDMs, apart from Scrum and XP. Quality Assurance and Configurations Management are weak areas in agile methods. Table 2.5 below tabulates the results of agile methods' contribution to acquiring process area maturity.

Table 2.5: Summary contribution value of agile methods in the fulfilment of process areas in maturity level 2 (Alegrıa and Bastarrica, 2006).

Area	Methods with major contribution	Fulfillment percentage (Capacity level + Percentage)
Requirement management	XP	CL-2 in 75%, CL-3 in 50%
Measurement and analysis	XP, ASD, Scrum	CL-2 in 75%, CL-3 in 50%
Project planning	XP, Scrum	CL-3 in 100%
Monitoring and control	XP, Scrum	CL-3 in 100%
Subcontract management	Not applicable	-
Product and process quality assurance	FDD, Crystal, Evo	CL-2 in 75%, CL-3 in 50%
Configuration management	none	CL-1 in 10%
Fulfillment percentage for maturity level 2		72%
Fulfillment percentage for maturity level 3		60%

The CMMi and Agile paradigms can be compared and contrasted along multiple dimensions. For the purposes of this research, a summary concept is presented to aid understanding of the viewpoints of both Agile and CMMi. Table 2.6 shows agile/CMMi comparisons and contradiction based on the criteria of organizational, management, planning, user assumptions, design, human development, Life-Cycle Emphasis and Predictability.

Table 2.6: Agile and CMMi comparisons and contradiction (Glazer et al 2008)

Dimension	CMMi Paradigm	Agile Paradigm
Organizational Focus	Focus is on the organization or enterprise. Beneficial when CMMi is implemented at organizational level so that all functions and capabilities contributing to the development of products/services are addressed by the process improvement effort.	Focuses on the project and team. ASDMs can separate the project/team from the organization and still be effective.
Management	Management is essential for ensuring project success. Emphasis is on project management, ensure that plans affecting the project are integrated within project plan, managed dependencies, coordination issues are resolved, there is a shared vision for the work, and risk management is performed.	Flexible management style that helps to eliminate barriers to progress. This view of management may be expanding as Agile approaches are extended to address larger project contexts.
Planning	CMMi promotes macro planning with an emphasis on establishing a suitable defined process enabling the project to achieve its objectives. CMMi does not require detailed planning but emphasises on re-planning and conditional change.	Multiple levels of planning that include high-level product planning. Beginning of iteration has detailed planning around the features to be addressed in that iteration. Strong emphasis on re-planning and flexibility as requirements change.
Market/User Assumptions	CMMi is beneficial when target market becomes more mature and process innovation is differentiating factor to organization success.	ASDMs strive most in an emergent and not well-understood target/market.
Perspective	CMMi assumes a longer period view.	ASDMs assume short to medium-term view.
Design Presumptions	CMMi presumes early product architecture selection and is revisited when it becomes clear the selected architecture is no longer valid or when using an iterative lifecycle	Projects are most successful when corporate standard architectures are adopted with flexibility applied as the project progresses.
Human	Project level has limited focus on people but expanded	ASDMs focus more on teams and individuals over

Development	people focus is at organizational level.	process.
Life-Cycle Emphasis	CMMi strongly emphasises on “review-as-you-develop”, encourages documentation, encourages frequent validations, analyses, and reviews before product components are integrated into a functional product.	ASDMs use parallel development, test iterations, and informal peer reviews of progress. “Fail early, fail fast, and learn” are central to Agile methods.
Cost to Failure	Cost of failure is high	Agile methodologies flourish in a domain of low cost of failure.

2.3.7 Summary

To summarise, it can be concluded that the co-existence of agile methodologies and CMMi in an organisation results in successful production of higher quality software, which more quickly satisfies the needs of the customers.

The next chapter describes how the research was conducted in terms of research paradigms, strategies, methods, tools and techniques.

CHAPTER 3

RESEARCH METHODOLOGY AND DESIGN

3.1 INTRODUCTION

This chapter discusses the research methodology used to achieve the objectives of the intended research. The chapter also describes the factors that influenced the selection methods of data collection and the limitations of the data that support the research. In order to achieve the primary aim, the research objectives that will assist in reaching the aim of this research are:

- Study the use and effectiveness of ASDMs in the industry.
- Study the use and effectiveness of SPIMs in the industry.
- Answer the question, does a relationship exist between systems development methodologies (SDMs) and accreditation with software process improvement models?

In an attempt to answer these questions, the researcher will identify companies in the information technology industry that are involved in software development.

The researcher opted for the positivistic research paradigm, and the survey as the research strategy. The survey research strategy is ideal to look for patterns and generalizations from sample standardized data of organisations. The use of the survey strategy aids in answering the research questions/objectives.

The quantitative data was generated from the questionnaires and needed to be analysed by applying a data-analysis tool. Quantitative data collected via questionnaires was analysed by applying tools such as SPSS version 16.

The chapter is structured in the following sequence of topics:

- Research paradigms.
- Research strategies.
- Data generation methods.
- Data analysis approaches.
- Summary

- Conclusion.

3.2 RESEARCH PARADIGMS

This section briefly identifies the various research paradigms available, and later describes in detail the research paradigm relevant to this research.

Research paradigms came into prominence in the 1980s. By the late 1980s, only the positivist and interpretive research paradigms had emerged and been extensively in use (Taylor, 2006). Critical social research came into existence only in the early 1990s, and its purpose is to consider other aspects such as social, cultural and political domination (Myers, 1997). According to Hopkins (2002), a research paradigm refers to those beliefs and methods that researchers utilise to explain the world around them.

The concept of "interpretivism" is believed to have a fuzzy philosophical background. This lack of a philosophical history makes the concept of interpretivism difficult to define. In its current use in social sciences and Information Systems research, it stands for a collection of research approaches that are best defined using their shared characteristic as being non-positivist. Interpretivism can thus be characterized by looking at the features of positivism that it discards and the alternatives suggested (Stahl, 2005).

The Interpretive research paradigm adopts the position that human knowledge of reality is a social construction of human actors. The researcher uses their own preconceptions in order to guide the process of data enquiry, therefore value-free data cannot be gathered. The researcher has direct interaction with fellow human subjects. This interaction is subject to change the perceptions of all parties involved in the research (Walsham, 1995).

In contrast to the positivist approach, Interpretivism assumes that the objective data collected by the researcher can be used to test prior hypotheses or theories.

The positivist approach, or scientific paradigm, is the oldest of the three research paradigms. The positivist approach refers to such procedures as those associated with hypothesis testing, mathematical analysis, inferential statistics, experimental and quasi-experimental design (Lee, 1991). In other words, it is a research approach associated with natural sciences.

This research adopts a positivist research paradigm. This paradigm was selected above

others because quantitative data would give a better indication of the uses - or lack of use - of SDMs and SPIMs in the industry. As a result, it would meet the research objectives as stipulated.

3.2.1 Positivist Research

The scientific research paradigm is moulded around two basic assumptions, namely: (a) The world is ordered and regular but not random; and (b) The world we live in can be investigated and interpreted in an objective manner (Oates, 2006).

Assumption (a) means that the manners in which events happen are ordered and often happen in a similar fashion, time and time again (that is, regularly). For example, Oates (2006:284) refers to a situation where an object rolls off the edge of table with absolute certainty that it will fall downwards. This will always occur. There is no random occurrence on planet Earth of gravity usually pulling objects downwards, but at other times pushing upwards instead, or occasionally even letting objects hover in mid-air.

Assumption (b) says that the world we live in, coupled with its regular laws and patterns, can be investigated objectively. This is as a result of the assumption that the laws and patterns exist independently of any individual's cognition. Therefore, personal feelings can be put aside and the researcher can be rational and objective in their interpretation of how the world operates.

The main objective of the scientific method is to establish universal laws, patterns and regularities. One research strategy commonly used in the scientific method is to carry out carefully-designed experiments to search for evidence that confirms or refutes the hypothesis (Oates, 2006). For example:

- **Hypothesis**: All zebras have black and white stripes.
- **Null** - Hypothesis: There exist zebras without black and white stripes
- **Test**: Investigate zebras and see whether they have black and white stripes.
- **Outcome**: All zebras seen had black and white stripes.
- **Conclusion**: Hypothesis is confirmed as true, that is, all zebras do have black and white stripes.

The scientific method has three basic techniques: repeatability, refutation and

reductionism.

- **Repeatability:** Researchers should not rely on a single outcome of a questionnaire. They should repeat the survey to ensure that the results of previous surveys were not by coincidence.
- **Refutation:** If other researchers cannot repeat the same survey and obtain the same results as the original survey, then the hypothesis can be proved false.
- **Reductionism:** Large organisations are broken down into IT departments small enough to be studied easily.

However, the scientific method is not suited to studying the "social world", that is, human beings and their interactions with their environment. Hence the paradigms often associated with the study of the social world are the Interpretive and Critical Social Paradigms.

3.2.1.1 Characteristics of the positivist approach

The positivism approach underlies the scientific paradigm. Researchers that adopt the paradigm mostly use experiments. However, as in this particular research, other research strategies such as surveys may be deployed in situations where experiments are not feasible (Oates, 2006). Oates identifies a number of characteristics of the positivist paradigm:

- **World is independent of humans**: The world's existence is not just a figment of our imagination. A physical and social world to be studied, captured and measured that exists in reality.
- **Measurement and modelling:** The world is investigated by observations and measurements and producing models of how it operates.
- **Objectivity**: The researcher ought to be neutral and objective, their role is to observe. Facts about the world can be deduced independently without the interference of the researcher's feelings and beliefs (Oates, 2008).
- **Hypothesis testing**: The research is based on the empirical testing of theories and hypotheses that concludes with either a confirmation or refutation of these hypotheses (Kim, 2003).
- **Quantitative data analysis**: There is a strong reliance on statistical analysis,

and mathematical modelling and proofs. Mathematics provides logical, objective way of analyzing observations and results. The researcher aims to establish a relationship between an independent variable and a dependent variable in a population (Matveev, 2002; Oates, 2008).

- **Universal laws**: The aim of the research is the generalization of universal laws, patterns or irrefutable facts that can hold true, regardless of the researcher or occasion.

3.2.1.2 Criticisms of Positivism

The scientific method or positivist approach is suitable for studying natural sciences. However, the world has its inhabitants - in the form of human beings who are a highly social species. The positivist approach is less suited to researching the social aspect of the world, that is, feelings and values people have, organizations and group structures that people build, cultures they develop, and so on:

- Repetition is not always possible.
- Reductionism, where complex things are broken down into smaller things, is not always possible.
- Generalization is not ideal in some instances.
- Individuals are unique and see the world in their own way.
- Regular laws and patterns may seem observable in the social world. However, these laws and patterns are in actual fact a construction of humans.

3.3 RESEARCH STRATEGIES

According to Myers (1997), a research strategy refers to a way of gaining insight into a particular phenomenon, and it is informed by the underlying epistemological assumption – either positivist, critical social, or interpretive. The research strategy of choice aids in moving from the underlying philosophical assumptions to research design and data collection. The research strategies commonly used are: Case studies, Action research, Ethnography and the Participant-Observation research strategy.

This section discusses the preferred Survey research strategy associated with the positivist research paradigm. A survey was a more suitable choice of strategy, as one of its strengths is to be able to provide generalizable statements concerning the subjects.

To further support the choice survey as research strategies, a summary of relative strengths of the strategies across several dimensions are provided in the Table below:

3.3.1 Survey research strategy

Survey research strategy is a quantitative method; it requires standardized information from and/or about the subjects being studied. The subjects studied might be individuals, groups, organizations, or communities; they also might be projects, applications, or systems (Pinsonneault & Kraemer, 1993).

The survey research strategy looks for patterns and generalizations from sample data. The main aim of surveys is to obtain the same kinds of data from a large group of people or events, in a standardized and systematic manner (Oates, 2006).

The researcher analyses the data generated from the survey to find patterns that can be generalized to a larger population than initially targeted.

It was critical that the survey conducted for this research remained anonymous, as the study required personal views of the employees with regards to the development processes and IT project success within their organization.

The target population for this research was software-developing organizations or individuals specialising in software development, systems analysis, Information Systems development, or project managers and relevant stakeholders. The survey was conducted over a period of 6 months (between October 2014 and March 2015) with the aid of a questionnaire as data collection method. The identified potential participants of the study were formally approached to be part of, and to recommend other IT professionals for their participation in the study. The formal approach was then followed by emailing the questionnaire package.

Over 500 emails were sent to organizations – both within our borders and far beyond, such as Pakistan, India, England and Zambia. Four months later, and after follow-up emails, only 35 questionnaires had been returned - a response rate of 7% per month. This return rate was quite low considering the time elapsed and targeted minimum sample of 120 questionnaires.

An online questionnaire was then developed using applications offered by Google documents. This allowed the respondents to follow a link and complete the survey

online. This method proved to be a success. Although the target sample of 120 questionnaires could not be reached in time, 100 questionnaires where sufficient for the study.

3.3.1.1 Advantages of using the Survey research strategy

Surveys study a representative sample of subjects such as organisations, and in doing so surveys seek to discover relationships that are common across organisations. Therefore, surveys provide generalizable statements concerning the subjects being studied (Gable, 1994). Surveys are able to identify extreme outcomes, document the norm accurately, and delineate associations between variables in a sample. Surveys provide wide coverage of people or events to represent the population (Oates, 2008) such that, via email, a large number of remote participants could be reached. Though it may have taken time to gather adequate data, the cost was considerably low (Oates, 2008).

3.3.1.2 Disadvantages of using the Survey research strategy

The survey strategy provides only a "snapshot" of the situation at a certain point in time; as a result, it yields little information on the underlying meaning of the data. Secondly, variables that are of interest to the researcher may not be measurable by the survey research strategy (Gable, 1994). Conducting surveys via fieldwork results in a poor method of objectively verifying the hypotheses.

3.4 DATA COLLECTION METHODS USED IN THIS STUDY

As noted earlier, the positivist research paradigm embraces the survey as a research strategy and questionnaire as a data collection/generation method (Mertens, 2005). However, other types of data generation methods exist, such as interviews, observations, and documents (Mertens, 2005; Oates, 2008).

This section discusses the data collection method that is adopted in this research. Although there are many different data generation methods, only questionnaires were deemed applicable to this research and thus are discussed in detail.

Data for this research was collected from primary sources. The study units comprised of software developing organisations, and information technology professionals such as project managers and senior software developers.

3.4.1 Questionnaires

Questionnaires are pre-defined sets of questions that are arranged in a pre-determined order. The use of questionnaires as data generation methods is frequently associated with survey research strategy, though it is not the only data generation method associated with surveys (Oates, 2006). Questionnaires may be self-administered or researcher administered. Self-administered questionnaires do not need the presence of a researcher to be completed, whilst researcher-administered questionnaires require that the researcher be present to note down the respondent's answers. In this research, self-administered, on-line and electronically mailed questionnaires were used.

3.4.1.1 Type of Questionnaires

- Contingency questions: Cater for only a sub-group of respondents and are regarded as a special case of closed-ended response (Ross, 2005).
- Open questions: This type of question allows respondents to respond to a question with an answer based on their understanding of the question (Weaver, 2005; Ross, 2005).
- Closed questions: These types of questions are fixed and allow the respondent to select their answer from a pre-defined set of answers (Ross, 2005).

3.4.1.2 Advantages of Questionnaires

Questionnaires were chosen as a data collection method. Several notable advantages led the researcher to adopt the use of questionnaires in the research (Milne, 1999):

- Questionnaires make it possible for information to be collected from a large portion of a group. This potential is not often realised, as returns from questionnaires are usually low;
- A questionnaire is an instrument that enables respondents to answer predefined questions;

- Data can be gathered in a standardised way, so questionnaires are more objective, certainly more so than interviews;
- The respondent can complete a questionnaire at a time convenient to them;
- A questionnaire empowers the respondent and protects their confidentiality;
- It is relatively quick to collect information using a questionnaire; and
- Questionnaires are more cost effective because they can be administered in groups or mailed to respondents.

These advantages of utilising a questionnaire were the driving factors in the adopting of the questionnaire as a data generation method. Questionnaires enabled reaching large and remote groups of respondents. However, as questionnaires posed the best fit of a data generation method, there are some disadvantages associated with the use of questionnaires. These disadvantages are highlighted below.

3.4.1.3 Disadvantages of Questionnaires

Given the advantages above, questionnaires do have disadvantages too as identified by Milne (1999):

- Some situations, questionnaires may take long to design, apply and analyse;
- Potential respondents may not be willing to answer questions if they not gaining anything from the survey or a fear the confidentiality of the information handed out;
- Questionnaires force the respondents to choose from a list of pre-selected answers that might not reflect their point of view;
- Questionnaires are standardised and thus pose potential misinterpretation of questions;
- Respondents may answer superficially, especially if the questionnaire is long and time-consuming to complete. The common mistake of asking too many questions should be avoided; and
- Open-ended questions can generate large amounts of data that can take a long time to process and analyse.

Inevitably, these disadvantages would impede the data-generation process. To

minimise the risk of the quality of the data collected, the researcher proposed several measures to be put in place, such as emphasis on confidentiality.

3.4.2 Questionnaire design

The questionnaire was designed to capture data that could be mined into useful information to satisfy the research objectives. A recap of the research objectives of this study are outlined below:

- Study the use and effectiveness of ASDMs in the industry.
- Study the use and effectiveness of SPIMs in the industry.
- Does a relationship exist between the use of systems development methodologies (SDMs) and software process improvement models (SPIMs).

The structure of the questionnaire is in such a manner that each section strives to capture information that directly or indirect answers the above research objectives. The four sections of the questionnaire are each presented below. Each section outlines the variables to be measured, the category of the variables and motivation/purpose of the question in the questionnaire.

3.4.2.1 Section A: Organisation background information

This section captured organisations' demographic information and IT department commitment towards certain activities. The purpose was to examine the nature of the environment in which the variables existed. The variables measured are tabulated in Table 3.1 below.

Table 3.1: Section A: Organization background information

Variables	Category	Motivation	Reference
Job category	4 options	Participants were asked to select their role in the organisation. it provides background information on the target population	Huisman & Iivari (2005)
Core business area	8 options to select one	Participants were asked to select the business area their organisation operated in. Provided background information on the business area of the organisation.	Huisman & Iivari (2005)
Organisational size	5 options to select one	Participants were asked to select a range which best describes the number of employees in the organisation. Provides information on the size of organisation	Huisman & Iivari (2005)
IS department size	5 options to select one	Participants were required to estimate the number of personnel in their department. This provides information on the size of the IS department.	Huisman & Iivari (2005)
New applications	Specify %	Participants were asked to specify percentage of IS department effort devoted to developing new applications. Provides information on what IS department concentrates on	Huisman & Iivari (2005)
Systems maintenance and support	Specify %	Participants were asked to specify percentage of IS department effort devoted to maintenance and support. Provides information on what IS department concentrates on	Huisman & Iivari (2005)
Package customization	Specify %	Participants were asked to specify percentage of IS department effort is devoted to package customization. Provides information on what IS department concentrates on	Huisman & Iivari (2005)
Project Size	5 options to select one	Participants were asked to indicate the size of the project they were last involved in.	Huisman & Iivari (2005)
Project duration	4 options to select one	Participants were asked to indicate duration of the project last involved in.	Huisman & Iivari (2005)

3.4.2.2 Section B: Systems development methodologies used

This section measures the degree of agile systems development methodology use, and the reasons for not using SDMs. The extent of use is measured on a five-point Likert scale ranging from 1: nominally used to 5: extensively used. Table 3.2 presents the variables found in this section.

Table 3.2: Section B: Agile systems development methodologies used

Variables	Category	Motivation	Reference
SDM & ASDM used	11 options to select multiple	Participants had an option to select multiple options and state the extent to which they used the methodology. This question aids in answering two objectives: (1) uses of ASDMS and (3) relationships between ASDMs and SPIMs uses.	Huisman & Iivari (2005)
Period of SDMs use	6 options to select one	Duration of methodology use has an effect on the project outcome. Contributes to answer objective 1.	Huisman & Iivari (2005)
Motivation factor for SDM choice	Motivating factors to choose one from or specify	Participants were asked to indicate the motivating factor for choosing a certain SDM. Provides information why organisations opt for a certain methodology. Contributes to answer objective 1	Huisman & Iivari (2005)
Proportion of projects using ASDMs	5 options to select one	Participants asked to indicate number of projects using. ASDMs. This indicates horizontal use of ASDMS in an organisation. Contributes to answer objective 1.	Huisman & Iivari (2005)
Proportion of employees applying ASDMs knowledge	6 options to select one	Participants asked to indicate proportion of personnel that follow ASDMs. This indicates vertical use of ASDMS in an organisation. Contributes to answer objective 1.	Huisman & Iivari (2005)
Severity of SDMs use	3 options to choose one	Participants were asked to select the extent to which they use ASDMs. This influences project outcome. Contributes to answer objective 1.	Huisman & Iivari (2005)
Current stance on ASDM use	3 options to choose one	Participants were asked to indicate the reason applicable to them with regards to not having an SDM in place. Provides information on project outcome and contributes to answering objective 1.	Huisman & Iivari (2005)
Factors for not using an SDM	13 factors with a Likert scale ranging from 1 (total disagreement) to 5 (total agreement.)	Participants had to indicate for each factor, the extent to which they thought each factor contributed to non-use of	Huisman & Iivari (2005)

		ASDMs. Affects objective 1.	

3.4.2.3 Section C: Software process improvement models

This section introduces software process improvement models. It is an integral part of the study as it helps in partial answering of the research objective 2 and 3. Table 3.3 shows the variables found in this section of the questionnaire.

Table 3.3: Section C: Software process improvement models

Variables	Category	Motivation	Reference
SPI certification	6 options to select one	Participants were given an option to select an SPI in use or specify. This data helps to answer the use of SPIMs (objective 2).	
CMMi maturity level	Specify CMMi maturity level	Participant asked to specify maturity level if certified with CMMi. Contributes to answer objective 2.	Ambler (2009)
Motivation for certification	1 out 5 Motivating factors to be certified.	Participants were asked to indicate the motivating factor for being certified. Provides data on why organisations choose to be certified. Contributes to answer objective 2.	
SDM influence of certification type	Yes/no	Participants asked whether the ASDM deployed had any influence in the type of certification in place. This question helps in find a relationship between the two approaches. Contributes to answering objective 3.	
Development processes and procedures	38 yes/no processes and procedures	Participants had to indicate which procedure and process they did. This helped to determine the class level at which they performed these activities. Contributes in answer-ing objective 2.	
Reason for non-certification	5 options to select 1 from	Participants were asked to indicate which reason best describes why organisation is not certified. Contributes to answer objective 2.	

3.4.2.4 Section D: Project outcome

This section describes the outcomes of projects embarked on by participants. The project outcome findings are essential to the study as they help to understand the use and effectiveness of both approaches under scrutiny. The variables in this section are presented in Table 3.4.

Table 3.4: Section D: Project Outcome

Variables	Category	Motivation	Reference
Project status	4 options to select one	Participants were given an option that describes the status of the project they were last involved in. This information helps to understand the outcome of projects and a links back to the uses of SDMs. Contributes to answer objective 1 and 2.	Huisman & Iivari (2005)
Duration of complete and running systems	Specify period in use	Participant asked to specify how long the system has been in use. Helps to answer objective 1 and 2	
Motivation for certification	1 out 5 Motivating factors to be certified.	Participants were asked to indicate the motivating factor for being certified. Provides data on why organisations choose to be certified. Contributes to answer objective 2.	
Project process success factors	9 influencing factors on a Likert scale ranging from 1 (total disagree) to 5 (total agree)	Participants asked to indicate the extent to which they agreed with factors affecting the process of developing the system. Contributes to answer objective 1 and 2.	Huisman & Iivari (2005)
Project product success factors	11 influencing factors on a Likert scale ranging from 1 (total disagree) to 5 (total agree)	Participants asked to indicate the extent to which they agreed with factors affecting the product. Helps in factor analysis of different factors affecting the final product. Contributes to answer objective 1 and 2.	Huisman & Iivari (2005)

3.4.3 The application of questionnaire strategies in the study

The study began with a need to investigate whether there exists a relationship between the use of agile systems development methodologies and software process

improvement models. A literature review on previous findings on ASDMs and SPIMs was conducted and then we adopted the positivist research methodology for further investigation on the topic. The survey was employed as a research strategy with the aid of a questionnaire.

This questionnaire (Appendix A) consisted of four sections of closed and open questions. The design of the questionnaire was mainly based on the instrument developed by Huisman and Iivari (2005). A mock trial of the questionnaire was sent for pre-testing to a systems analyst, a senior programmer and a project leader. The response showed that some of the longer closed questions were incomplete and the open questions requesting their opinions were left blank. The questionnaire was adjusted to minimise the number of open questions and removed closed questions that were deemed non applicable to the study.

The questionnaire targeted personnel working in the IT departments where there was some form of software development. The job titles of the target personnel preferred were project managers/leaders, senior developers, or systems analyst.

The questionnaire was distributed to a wide spectrum of IT professionals regardless of country base. The decision to diversify the research was to allow the researcher to uncover the general perceptions of deploying ASDMs and SPIMs across the spectrum of an organisation.

A ‘request for participation’ letter was emailed with either the attached questionnaire or with the link to the online form. The letter introduced the researcher and provided a brief background to what the study is all about. It also assured the recipient of confidentiality of any responses.

The 35 email responses were manually added to the 65 received via the online form spread sheet. This allowed easy data coding in readiness for data analysis.

3.5 DATA-ANALYSIS METHODS USED IN THE STUDY

The positivist research paradigm adopted resulted in the generation of quantitative data. The use of surveys as a research strategy resulted in data, or evidence that is expressed in numerical terms. This type of data is referred to as **quantitative data**. This type of data can be produced by other strategies too but it is prominent in experiments

and surveys (Oates, 2006). Data analysis is ideally a means of finding patterns in the data captured and drawing up conclusions. Quantitative data can be analysed by various techniques such as tables, graphs and charts.

Once the data had been collected, the validity of the questionnaire had to be tested. The validity of a questionnaire can be described as the degree to which a question actually measures what it is supposed to test (Ross, 2005).

The data analysis process was made possible by acquiring the services of a statistics expert, Dr Suria Ellis from the North West University, Potchefstroom, South Africa. Using Statistical Package for the Social Sciences (SPSS), Dr Ellis validated the generated data by using data analysis techniques such as (Bryman & Cramer, 2002):

- Factor analysis: To describe variability between observed and or correlated variables in terms of a possibly lower number of unobserved variables referred to as factors (Matsunaga, 2015).
- Contingency tables (also called crosstabs): Are used to summarize the relationship between two independent variables, for example, crosstabs of each ASDM x and SPIM y. The contingency table will show the number of times x and y combinations occurred in the population sample.
- Descriptive statistics: Used to define the collected data from the questionnaires by providing basic summarizations on the sample and the measures.
- Correlation analysis: This technique is also used to investigate the relationship between two quantitative, continuous variables.
- Frequency table: Table showing regularity counts for categorical variables.

3.6 DATA-ANALYSIS TOOLS

In this section, tools used in analysing the data are discussed. The main tool identified for this purpose was SPSS version 16. Below is a brief description of the above - mentioned data analysis tool.

3.6.1 SPSS version 16.0

SPSS is a statistical quantitative data analysis. First released in 1968, SPSS allows the

researcher to use statistical principles and formulae in order to identify patterns and similarities between the cases being studied. Once the minimum number of questionnaires had been collected from the research subjects, the data was compiled and coded on one spread-sheet into Excel. The coding process involves transforming the inputs into numerical values for closed questions, and text inputs for open questions.

3.7 SUMMARY

This chapter discussed the research methodology and design which this study follows to achieve its primary objectives. As an introduction to general research methodologies, a brief description of the different types of research paradigms being critical, positivist and interpretive have been outlined.

The positivist research paradigm was adopted. The survey research strategy was identified as the strategy to follow in the study, and as such it is discussed in the chapter. The questionnaire was endorsed as a data generation method and as a result generated quantitative data.

Quantitative data analysis with aid of data analysis tools such as SPSS was conducted in a bid to: (1) Answer the research objectives, (2) provide an academic contribution to the topic and, (3) provide practical assistance to software houses seeking accreditation. It is through the adoption of the recommended research methodology that these objectives can be investigated. The next chapter presents the data that was collected from the questionnaires, as well as results generating from the application of data analysis tools and techniques.

CHAPTER 4

RESEARCH RESULTS

4.1 INTRODUCTION

This chapter analyses and interpret the data gathered to give insight to the empirical study conducted. The results are presented using statistical analysis through the following tests that were performed on the sample data collected (Bryman & Cramer, 2002):

- Frequency tables
- Factor analysis
- Descriptive statistics,
- Crosstabs: The crosstabs will also include effect sizes as a result of working with a convenience sample, and
- Correlations

In addition, as the size of the data set increases, there is a tendency to yield small p-values when applying significance tests. The results yielded from crosstabs and correlations also indicate the effect size. The effect size does not depend on the size of the sample but is rather a measure of practical significance. It is considered a large enough effect that is important in practice. Many effect sizes are found, such as those describing correlation between two variables, for multiple regression and for the difference between means (Ellis & Steyn, 2003).
The researcher had decided to make use of questionnaires as a data collection tool, because of the remoteness of potential respondents relative to where the researcher was based.

Most of the Primary data was collected via an online questionnaire sent to potential respondents in countries worldwide, such as the United Kingdom, South Africa and Zambia (to mention a few). The research focused on the organisational use, and existence (if any) of a relationship between ASDMs and SPIMs, therefore the geographical location of the organisation or respondents was considered to be irrelevant.

The ideal target respondents for the research were professionals in the information technology industry such as IT Managers, Software Developers, and Systems Analysts.

Over 500 emailed and dropped-off questionnaires were distributed, and only about 35 responded. A change in strategy was implemented by creating a Google Form-based questionnaire that would allow the respondents to simply click on an option as opposed to writing/typing answers. The online questionnaire proved to be a better method to get responses, as it resulted in 65% of the responses.

4.2 RESEARCH AIMS AND OBJECTIVES

Before presenting the results of the statistical analysis, it is important to revisit the research objectives. The objectives of this research are:

- Study the use and effectiveness of ASDMs in the industry.
- Study the use and effectiveness of SPIMs in the industry.
- Investigate the existence (or lack of) a relationship between ASDMs and SPIMs, with regards to their use in the IT industry.

A total 100 questionnaires were collected from the different organisations, and these were analysed using SPSS version 16.0. Each question that appears in the questionnaire will be discussed separately. There are four sections in the questionnaire, dealing with Organisation Background Information, Systems Methodologies Used, Software Process Improvement Models Certification, and Project Outcome. These results of each section are discussed separately, as follows.

4.3 SECTION A: Organisation Background Information

This section aimed to collect background information of the organisation and the respondent. This included job responsibilities of the respondent, business area of the organisation, organisation size, and (in particular) data on the information systems department functionality.

4.3.1 Job category

The questionnaire provided four options from which the respondents could choose a job category describing their job title within the organisation. The options included Project Manager/Leader, Software Developer, and Systems Analyst. The fourth option allowed them to specify a position that was not covered by the first three options.. The results

were as follows: 23 out of 100 of those respondents were Project Managers/Leaders. Most of the respondents (52) were Software Developers. Systems Analysts represented 13 out of 100 of the respondents. Table 4.1 shows actual results of respondents' job descriptions.

Table 4.1: Job category

Frequency Table: Job descriptions			
		Frequency	Valid Percent
Valid	Project Manager	23	23.0
	Developer	52	52.0
	Systems Analyst	13	13.0

However, the respondents could choose more than one job category. The fourth option, where the respondents specified a job category, had 20 responses. Table 4.2 presents the specific job categories specified by the respondents.

Table 4.2: Specified Job category Job category

Frequency Table: Other job descriptions					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		92	47.9	47.9	47.9
	0	80	41.7	41.7	89.6
	Agile Coach	1	.5	.5	90.1
	Analyst Programmer	1	.5	.5	90.6
	Associate Security Engineer	1	.5	.5	91.1
	BI Architect	1	.5	.5	91.7
	Business Analyst	2	1.0	1.0	92.7
	CEO	1	.5	.5	93.2
	Chief architect	1	.5	.5	93.8
	Consultant	1	.5	.5	94.3
	IT specialist	1	.5	.5	94.8
	IT manager	1	.5	.5	95.3
	IT Manager	1	.5	.5	95.8
	IT strategy and Transformation manager	1	.5	.5	96.4
	Network Engineer	1	.5	.5	96.9
	Process Consultant	1	.5	.5	97.4
	Software Tester	1	.5	.5	97.9
	Systems Administrator	1	.5	.5	98.4
	Systems Administrator	1	.5	.5	99.0
	unknown	1	.5	.5	99.5
	unspecified	1	.5	.5	100.0
	Total	192	100.0	100.0	

4.3.2 Core business area

A total of 54% responses represented Systems Development as a core business area of their organisation. Another 15% represented the Financial Banking sector. Organisations involved in Administrative Services and Manufacturing represented 3% of the respondents each. The Education sector and Government got the least responses of 2% each. A total of 21% of the respondents specified different core business areas. Table 4.3 shows the results of the organisations' core business area.

Table 4.3: Organisation core business area

Frequency Table: business core area					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Administrative Services	3	1.6	3.0	3.0
	Manufacturing	3	1.6	3.0	6.0
	Education	2	1.0	2.0	8.0
	System Development	54	28.1	54.0	62.0
	Financial Banking	15	7.8	15.0	77.0
	Government	2	1.0	2.0	79.0
	Other Area	21	10.9	21.0	100.0
	Total	100	52.1	100.0	
Missing	System	92	47.9		
Total	192	100.0			

The Agriculture and Telecommunications sectors returned 3% each of the responses, whilst IT Infrastructure Service Providers and Mining provided 2% each. The other sectors - such as communications, media/broadcasting and transport - had single respondents. Table 4.4 presents the results from the specified core business area by respondents.

Table 4.4: Other organisation core business area

Frequency Table: Other business areas					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		171	89.1	89.1	89.1
	Agriculture	3	1.6	1.6	90.6
	Broadcasting\media	1	.5	.5	91.1
	Communications	1	.5	.5	91.7
	Information Systems Freelance	1	.5	.5	92.2
	Insurance	1	.5	.5	92.7
	Insurance & Financial Services	1	.5	.5	93.2
	IT Infrastructure Service Provider	2	1.0	1.0	94.3
	Legal Software	1	.5	.5	94.8
	Media/Broadcasting	1	.5	.5	95.3
	Mining	2	1.0	1.0	96.4
	System Integration/Security	1	.5	.5	96.9
	Telecommunications	3	1.6	1.6	98.4
	Telecoms	1	.5	.5	99.0
	Transport	1	.5	.5	99.5
	Travel and Tourism	1	.5	.5	100.0
	Total	192	100.0	100.0	

4.3.3 Organizational size

Forty-three percent of the respondents were from organisations with more than 1,000 employees. Thirty-one percent had a maximum of 50 employees in the organisation. Organisations falling between the “51-200” employees range had 17 respondents, while 6% of the respondents said their organisation employed between 201 and 500 people. The fewest respondents (with only 3%) fell between the “501 and 1000” range. Table 4.5 shows the actual results of the staff-complement ranges.

Table 4.5: Total number of employees in an organisation

Frequency Table: Organisational size					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1-50 employees	31	16.1	31.0	31.0
	51-200 employees	17	8.9	17.0	48.0
	201-500 employees	6	3.1	6.0	54.0
	501-1000 employees	3	1.6	3.0	57.0
	More than 1000 employees	43	22.4	43.0	100.0
	Total	100	52.1	100.0	
Missing	System	92	47.9		
Total		192	100.0		

4.3.4 Information systems departmental size

Forty-two percent of the respondents indicated that there were more than 50 employees in their IS department. The second-most responses fell in the '20 to 50' range which had 25% of the respondents. The rest of the respondents worked in IS departments with fewer than 20 employees. Table 4.6 displays the results of the size IS department within the organisation.

Table 4.6: Information Systems department size

Frequency Table: IS department size					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1-5 employees	19	9.9	19.0	19.0
	6-20 employees	14	7.3	14.0	33.0
	20-50 employees	25	13.0	25.0	58.0
	More than 50 employees	42	21.9	42.0	100.0
	Total	100	52.1	100.0	
Missing	System	92	47.9		
Total		192	100.0		

4.3.5 IS investment in new applications

A total of 17.7% of the respondents spent 50% of their time developing new applications in their IS department, while 5.2% devoted most of their time to new application development. However, 2.1% did not spend any time on developing new applications. Table 4.7 shows the distribution of the time spent on developing new applications.

Table 4.7: IS department's effort to development of new applications

Frequency Table: New applications investment					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.0%	2	1.0	2.1	2.1
	1.0%	1	.5	1.0	3.1
	5.0%	3	1.6	3.1	6.3
	10.0%	5	2.6	5.2	11.5
	15.0%	1	.5	1.0	12.5
	20.0%	10	5.2	10.4	22.9
	25.0%	3	1.6	3.1	26.0
	30.0%	9	4.7	9.4	35.4
	35.0%	1	.5	1.0	36.5
	40.0%	12	6.3	12.5	49.0
	50.0%	17	8.9	17.7	66.7
	60.0%	8	4.2	8.3	75.0
	65.0%	6	3.1	6.3	81.3
	70.0%	2	1.0	2.1	83.3
	75.0%	3	1.6	3.1	86.5
	80.0%	6	3.1	6.3	92.7
	90.0%	2	1.0	2.1	94.8
	100%	5	2.6	5.2	100.0
	Total	96	50.0	100.0	
Missing	System	96	50.0		
Total		192	100.0		

4.3.6 IS investment into maintenance and support

Seventeen-and-half percent (17.5%) of the respondents put 40% of their efforts into systems maintenance and support; 12.4% put in more effort at 60%; while 6.2% said

they devoted all their time to systems maintenance and support. Only 1% of the respondents did not invest any time in systems maintenance and support. Table 4.8 shows the distribution of the time spent on systems maintenance and support.

Table 4.8: IS department's effort on systems maintenance and support

Frequency Table: IS effort put into maintenance and support					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.0	1	.5	1.0	1.0
	5.0%	1	.5	1.0	2.1
	10.0%	3	1.6	3.1	5.2
	15.0%	1	.5	1.0	6.2
	18.0%	1	.5	1.0	7.2
	20.0%	11	5.7	11.3	18.6
	25.0%	6	3.1	6.2	24.7
	30.0%	13	6.8	13.4	38.1
	35.0%	3	1.6	3.1	41.2
	40.0%	17	8.9	17.5	58.8
	50.0%	8	4.2	8.2	67.0
	60.0%	12	6.3	12.4	79.4
	65.0%	1	.5	1.0	80.4
	70.0%	4	2.1	4.1	84.5
	75.0%	1	.5	1.0	85.6
	80.0%	4	2.1	4.1	89.7
	90.0%	4	2.1	4.1	93.8
	100.0%	6	3.1	6.2	100.0
	Total	97	50.5	100.0	
Missing	System	95	49.5		
Total		192	100.0		

4.3.7 IS investment to package customization

The majority of respondents (20.2%) invested only about 10% of their time to customizing their product. A mere 3.6 % devoted all their time, whilst 9.5% did not attempt to customize their product. Table 4.9 illustrates the results of questionnaire responses to the efforts put into package customization by the IS department.

Table 4.9 : IS department's effort to package customization

Frequency Table: IS effort put to package customization					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.0%	8	4.2	9.5	9.5
	2.0%	1	.5	1.2	10.7
	5.0%	5	2.6	6.0	16.7
	10.0%	17	8.9	20.2	36.9
	15.0%	5	2.6	6.0	42.9
	20.0%	13	6.8	15.5	58.3
	25.0%	7	3.6	8.3	66.7
	30.0%	7	3.6	8.3	75.0
	40.0%	3	1.6	3.6	78.6
	45.0%	1	.5	1.2	79.8
	50.0%	8	4.2	9.5	89.3
	60.0%	1	.5	1.2	90.5
	70.0%	2	1.0	2.4	92.9
	80.0%	2	1.0	2.4	95.2
	90.0%	1	.5	1.2	96.4
	100.0%	3	1.6	3.6	100.0
	Total	84	43.8	100.0	
Missing	System	108	56.3		
Total		192	100.0		

4.3.8 Last IS project size

Most of the respondents (43 out of 100, namely 43%) had been involved in large projects. Medium projects had occupied another 27%. Those involved in small projects accounted for 11%. Very large projects attracted about 17% of the respondents whilst only 2% did not partake in any project. Table 4.10 displays the results of the last project

size in which respondents were involved.

Table 4.10: Size of IS department's last project

Frequency Table: Last project size					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Small	11	5.2	11.0	11.0
	Medium	27	14.1	27.0	37.0
	Large	43	22.4	43.0	80.0
	Very Large	17	8.9	17.0	97.0
	Non partaken	2	1.0	2.0	99.0
	Total	100	52.1	100.0	
Missing	System	92	47.9		
Total		192	100.0		

4.3.9 Duration of last project

Most projects undertaken took less than a year to complete; 47% of the respondents indicated that they took less than a year. Another 37 percent took between a year and two years to finish. Thirteen percent of respondents took between three and five years to complete the project. Only 3% of the projects took longer than six years. Table 4.11 below displays the results of the project durations.

Table 4.11: Duration of last project

Frequency Table: Duration of last project					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Less than a year	47	24.5	47.0	47.0
	1-2 years	37	19.3	37.0	84.0
	3-5 years	13	6.8	13.0	97.0
	6 or more years	3	1.6	3.0	100.0
	Total	100	52.1	100.0	
Total		192	100.0		

4.4 SECTION B: Systems development methodologies used (Research objective 1)

This section of the questionnaire focused on the types of, and the extent to which systems development methodologies are used within the organisation. Here we unravel the answers to one of the study's objectives (objective 1) being "the use and effectiveness of ASDMs in the industry".

4.4.1 Commercial systems development methodology usage

The respondents were given an option to indicate on a Likert scale (1 to 5), where a 1 represents nominal use of the methodology, 2 represents low use, 3 is average use, 4 represents high use, and intensive use is represented by a 5. The results of each SDM's usage are discussed below and shown in Table 4.12.

- ***Systems Development Life Cycle (SDLC)***

  Of the 88 responding to the use of the SDLC, 49 of the respondents (55.7%) generally had a high usage of SDLC methodology. Twenty-five percent indicated that they seldom used the methodology. Nearly 20% had an average use of SDLC.

- ***Information Engineering (IE)***

  The study found a very low usage of IE methodology by the respondents, as indicated by 54.7% of them saying that they only nominally used the methodology.

Only 20.4% of the respondents generally used IE methodology intensely. The rest of the 25% averagely used the methodology.

- ***Rational Unified Process (RUP)***

  A large percentage (68.8%) had a nominal-to-low usage of the RUP methodology, signifying a very low use of the methodology by the respondents.

- ***Scrum***

  Scrum methodology also scored a low usage figure amongst the respondents as 50.6% indicated that they rarely deployed Scrum methodology. Close to 28% used the methodology averagely and 26.6% had a high use of the Scrum methodology

- ***Rapid Application Development (RAD)***

  The methodology seems to have no overwhelming use, as 37.8% of the respondents generally indicated that they rarely used the methodology, while 39.2% showed that they used it often.

- ***Extreme Programming (XP)***

  In this category, 48 out of 74 respondents (64.8%) indicated minimal use of XP, signifying that the methodology was rarely implemented by the respondents.

- ***Lean Software Development (LSD)***

  In this category, 74.6% of the respondents generally did not make use of LSD, signifying that LSD had low usage by the respondents.

- ***Feature-Driven Development (FDD)***

  FDD also generally had low usage (56.1) as indicated by the respondents (37 out of 66). Another 25.8% indicated that the methodology was averagely used. Only 18.2% revealed high FDD usage.

- ***Crystal Methodologies***

  Most of the respondents (72.6%) revealed low Crystal Methodology usage, signifying that the methodology is not commonly used.

- ***Dynamic Systems Development Methodologies (DSDM)***

  Similarly to Crystal Methodologies, there is low usage of DSDM, as shown by the results from the respondents (72.6%).

- ***Adaptive Software Development (ASD)***

  Fewer than 20% of the respondents indicated a high positive use of ASD, whilst 67.2% of the respondents rarely used the methodology.

- ***Other, please specify***

  The respondents also had an option to specify a type SDM being used in their organisations. Fifty-two percent of the respondents (26 out of 50) indicated that other methodologies were highly used; 26% indicated a low use; and 22 percent said they averagely used the methodology.

Table 4.12: Commercial SDMs usage rate in %

Frequency Table: Commercial SDMs usage rate												
	Nominally		Low		Average		High		Intensively		Totals	
	Freq	%	Freq	%	Freq	%	Freq	%	Freq	%	Total Freq	Freq Mean
Systems Development Life Cycle (SDLC).	14	15.9	8	9.1	17	19.3	21	23.9	28	31.8	88	3.47
Information Engineering (IE)	23	35.9	12	18.8	16	25	9	14.1	4	6.3	64	2.36
Rational Unified Process	30	46.9	14	21.9	11	17.2	9	14.1			64	1.98
Scrum	22	29.3	16	21.3	17	22.7	10	13.3	10	13.3	75	2.60
Rapid Application Development (RAD)	14	18.9	14	18.9	17	23	18	24.3	11	14.9	74	2.97
Extreme Programming (XP)	20	37.8	28	27	14	18.9	7	9.5	5	6.8	74	2.31
Lean Software Development	35	55.6	12	19	11	17.5	3	4.8	2	3.2	63	1.81
Feature-Driven Development	30	45.5	7	10.6	17	25.8	10	15.2	2	3.0	66	2.20
Crystal Methodologies	41	65.1	8	12.7	8	12.7	3	4.8	3	4.8	63	1.71
Dynamic Systems Development Methodologies	37	59.7	8	12.9	12	19.4	3	4.8	2	3.2	62	1.79
Adaptive Software Development (ASD)	34	53.1	9	14.1	9	14.1	10	15.6	2	3.1	64	2.02
Other, please specify	7	14	6	12	11	22	10	20	16	32	50	3.44

4.4.2 Period of systems development methodology usage

The question required the respondents to indicate how long the SDMs had been in use in their Information Systems departments. Of the questionnaires collected, 96 respondents indicated the duration of the SDM in their IS department. It was found that 9.4% of the respondents (9 out of 96) have had their SDM for less than a year; 48% fell in the '1 to 5 years' range. Another 13.5 percent of the respondents did not know how long the SDM had been in use. Table 4.13 shows the results for question 2.

Table 4.13: Duration of SDM use in IS department.

Frequency Table: Duration of SDM use					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Less than a year	9	4.7	9.4	9.4
	1 to 2 years	16	8.3	16.7	26.0
	3 to 5 years	30	15.6	31.3	57.3
	6 to 10 years	19	9.9	19.8	77.1
	11 years or more	9	4.7	9.4	86.5
	Unknown	13	6.8	13.5	100.0
	Total	96	50.0	100.0	
Missing	System	96	50.0		
Total		192	100.0		

4.4.3 Motivation systems development methodology choice

The statements in this question measured the factors that motivated the choice of the SDMs, such as productivity, software process certification, and process standardisation. Below the results of this measure are discussed and shown in Table 4.14.

- ***Increased productivity and quality***

  A total of 45.8% of the respondents (44 out of 96) indicated that the statement was a motivating factor for the choice of SDMs in their IS department. Therefore, increased productivity was the leading motivating factor for the choice of SDMs.

- ***Software process improvement certification***

  Only three of the respondents (3.1%) argued that the need for software process improvement certification was the motivating factor for the choice of SDM.

- ***Standardizing the development process***

  Some 35.4% of respondents (34 out of 96) were motivated by the need to standardize the development process when choosing the SDM in their IS department. The rest (15.6%) did not know the reason/motivating factors for choosing an SDM.

Table 4.14: Motivation for SDM choice

Frequency Table: Motivation for SDM choice					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Increased productivity and quality	44	22.9	45.8	45.8
	Software process improvement certification	3	1.6	3.1	49.0
	Standardizing the development process	34	17.7	35.4	84.4
	Unknown	15	7.8	15.6	100.0
	Total	96	50.0	100.0	
Missing	System	96	50.0		
Total		192	100.0		

4.4.4 The proportion of projects developed by using systems development methodologies

This question asked respondents to indicate the proportion of projects that had been developed by applying SDMs knowledge. Thirty seven out of 96 respondents (38.5%) answered that over 75% of their projects were developed by applying SDM knowledge.

Only 6.3% of the respondents did not apply SDMs in developing their projects. The remaining 53 respondents applied up to 75% SDM knowledge to system development. Table 4.15 shows the detailed results of the proportion of projects developed by using systems development methodologies knowledge.

Table 4.15: Proportion of projects developed with aid of SDMs

Frequency Table: Proportion of project by SDMs					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	None	6	3.1	6.3	6.3
	1-25%	12	6.3	12.5	18.8
	26-50%	21	10.9	21.9	40.6
	51-75%	20	10.4	20.8	61.5
	More than 75%	37	19.3	38.5	100.0
	Total	96	50.0	100.0	
Missing	System	96	50.0		
Total		192	100.0		

4.4.5 Proportion of employees who used systems development methodology regularly

This question measures the number of people that employed system development methodologies. Over 30% of respondents answered that over 76% of the employees in the IS department applied SDM knowledge on a regular basis. Another 46.3% of the respondents said between 50 and 75% of the people applied SDM knowledge. However, 4.2% of the respondents said none of the employees applied SDM knowledge. Table 4.16 shows the proportions of employees that apply SDM knowledge.

Table 4.16: Proportion of people applying regular SDM knowledge in IS department.

Frequency Table: Proportion of employees using SDMs					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	None	4	2.1	4.2	4.2
	1-25%	18	9.4	18.9	23.2
	26-50%	21	10.9	22.1	45.3
	51-75%	23	12.0	24.2	69.5
	More than 76%	29	15.1	30.5	100.0
	Total	95	49.5	100.0	
Missing	System	97	50.5		
Total		192	100.0		

4.4.6 Perceived use of systems development methodologies

The statements in this question aimed to describe how the IS department made use of SDMs. The results gathered are discussed per statement and shown in Table 4.17 below.

- *A Standard which is followed rigorously for all projects*
  A total of 27.7% of the respondents (26 out of 94) agreed that SDMs were used as a standard to follow thoroughly for all projects.
- *A general guideline for all projects*
  A total of 38.3%of the respondents (36 out of 94) preferred to use SDMs as a guideline for all projects. This statement best described how IS department made use of SDMs.
- *Adapted on a project-to-project basis*
  Exactly 34% of the respondents said that they adapted their use of SDMs on a project-to-project basis.

Table 4.17: Perceived use of SDMs

Frequency Table: Perception on SDM use					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	A Standard which is followed rigorously for all projects	26	13.5	27.7	27.7
	A general guideline for all projects	36	18.8	38.3	66.0
	Adapted on a project-to-project basis	32	16.7	34.0	100.0
	Total	94	49.0	100.0	
Missing	System	98	51.0		
Total		192	100.0		

4.4.7 Sub-section B: Reasons for not using systems development methodologies

This section establishes the reasons for not deploying systems development methodologies in the development of information systems.

4.4.7.1 Experience with the interaction of SDMs

The intention of this question being included in the questionnaire was to capture a sense of the experience that prompted the organisation not to use SDMs. A total of 31 respondents answered this question. The results yielded from the three statements are briefly discussed below - and tabulated in Table 4.18, and graphically presented in Figure 4.1.

- *IS department had never considered using SDMs*
  Fourteen out of 31 respondents stated that they had not considered using SDMs in their IS department.
- *IS department had considered using SDMs, but decided against it*
  Ten of the respondents stated that the IS department had considered using SDMs, however for some reason(s) had decided against it.
- *IS department had used SDMs in the past, but abandoned them*

Seven respondents answered that they had previously made use of an SDM, however the SDM has been abandoned.

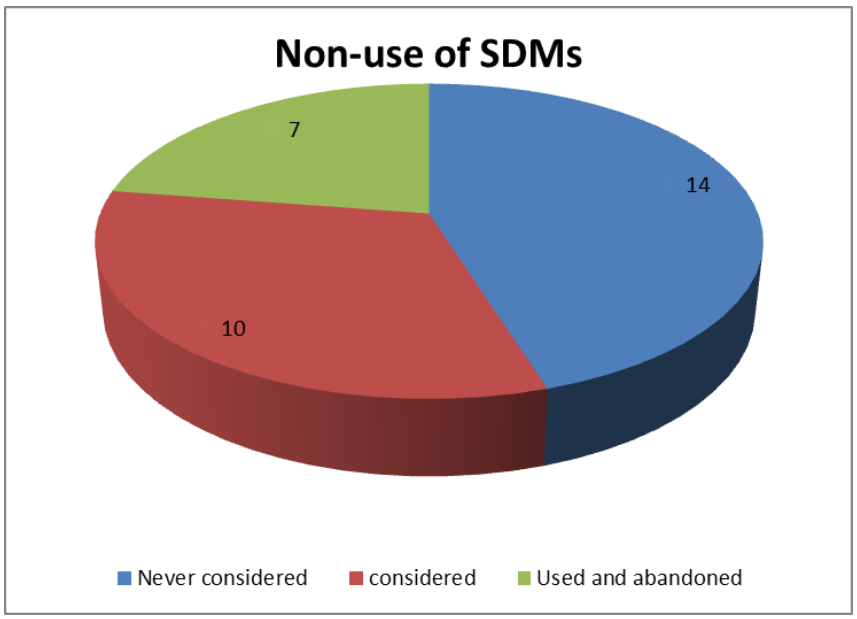

Figure 4.1: Experiences leading to non-use of SDMs

Table 4.18: Rate of experience of non-SDM usage

Frequency Table: Non-use of commercial SDMs					
Category		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Never used SDMs	14	7.3	45.2	45.2
	Considered using SDMs but decided against it.	10	5.2	32.3	77.4
	Used SDMs in the past, but abandoned them	7	3.6	22.6	100.0
	Total	31	16.1	100.0	
Missing	System	161	83.9		
Total		192	100.0		

4.4.7.2 Reasons for not adopting SDMs in last project

These statements assessed the successful (or lack thereof) completion of the last project the respondents were involved with, without using SDMs. Table 4.19 contains

the generalised frequencies and percentages of the reasons for non-SDM use. Each reason's results will be discussed

The most prevalent reason why systems development methodologies were not used is the one with the lowest mean in Table 4.32. It is clear that benefits of SDM use are long-term, whereas costs are incurred short-term, and the least- mentioned aspect reflects that the profile of development projects in the IS department does not require the use of SDMs.

- *The profile of development projects in IS department doesn't require use of SDMs*
  A total 63.3% of the respondents (24 out of 49) generally disagree with the statement. Slightly above 20% generally agreed, and the other 16.3% remained neutral.
- *SDMs are too complex or hard to use*
  Respondents generally disagreed with the notion that SDMs are complex to implement (65.3%). Some 18.3% of the respondents did agree with the statement.
- *The current systems development practice in our IS department is adequate*
  Only 20.4% disagreed that the current development practices were adequate. Meanwhile, the majority (46.9%) agreed with the statement and 32.7 percent assumed a neutral position.
- *The experience of the developers in IS department reduces need for SDMs*
  The respondents seemed to be undecided on this statement, as 37.5% generally disagreed while 35.5% generally agreed; 27.1% remained neutral.
- *The benefits of SDMs use are long-term, whereas costs are incurred short term.*
  A total of 15.2% of the respondents generally disagreed with the statement. The majority were neutral while 36.9% generally agreed.
- *Lack of experienced staff in IS department who can effectively use SDMs.*
  Fifty-one percent of the respondents disagreed that lack of staff experience may be a contributing reason.
- *New systems developed with SDMs not compatible with legacy systems*
  Compatibility with legacy systems appeared to be not an issue, as 50% generally disagreed with it being the reason behind not adopting SDMs.
- *IS department lacks a suitable environment to support SDMs*

The majority of the respondents dismissed lack of suitable environment, as 50.6% generally disagreed. The remaining respondents were either neutral or somewhat agreed.

- *IS department lacks management support for use of SDMs*
  An overwhelming 66.7% disagreed that there was lack of management support for SDM use.
- *Long learning curve for SDMs*
  A total of 33.3% cited that the learning curve for SDMs was too long, 42.2% disagreed and the rest (24.4%) were undecided and chose neutrality.
- *The financial investment in SDMs is too large*
  Lack of finances to invest in SDMs was chosen as the reason by 15.2% of respondents, with 43.5% generally disagreeing that this was the cause for non-SDMs use.
- *In our IS department there is a lot of uncertainty over the benefits of adopting systems development methodologies*
  A total of 44.6% of the respondents disagreed that uncertainty of the benefits of adopting SDMs would be a reason why organisations did not adopt SDMs, whilst only 17.1% agreed with the reason.
- *IS department has no clear objectives for adopting SDMs*
  A total of 51.1% of respondents generally disagreed with lack of clear objectives as a cause for not using SDMs. Another 21.3% of the 10 that responded to this question agreed that there were no clear objectives for adopting SDMs.

Table 4.19: Reasons for non-SDMs usage %

Frequency Table: Reasons for non-use of SDMs								
Reasons for non SDM usage	Generally disagree		Neutral		Generally agree		Totals	
	Freq	%	Freq	%	Freq	%	Total Freq	Total %
The current systems development practice in our IS department is adequate.	10	20.4	16	32.7	23	46.9	49	100
The benefits of SDMs use are long-term, whereas costs are incurred short term.	7	15.2	22	47.8	17	36.9	46	100
The experience of the developers in IS department reduces need for SDMs	18	37.5	13	27.1	17	35.5	48	100
Long learning curve for SDMs	19	42.2	11	24.4	15	33.3	45	100
Lack of experienced staff in IS department who can effectively use SDMs	24	51	10	21.3	13	27.6	47	100
IS department lacks a suitable environment to support SDMs	27	56.3	10	20.8	11	23	48	100
IS department has no clear objectives for adopting SDMs	24	51.1	13	27.7	10	21.3	47	100
The profile of development projects in IS department doesn't require use of SDMs	31	63.3	8	16.3	10	20.4	49	100
IS department lacks management support for use of SDMs	32	66.7	7	14.6	9	18.8	48	100
SDMs are too complex or hard to use	32	65.3	8	16.3	9	18.3	49	100
In our IS department there is a lot of uncertainty over the benefits of adopting systems development methodologies	21	44.6	18	38.3	8	17.1	47	100
The financial investment in SDMs is too large	20	43.5	19	41.3	7	15.2	46	100
New systems developed with SDMs not compatible with legacy systems	24	50	18	37.5	6	12.6	48	100

4.4.7.3 Factor analysis and reliability

In this section, factor analysis is presented; using un-rotated factor loadings to factors which describe the reasons for non-use of systems development methodologies.

Table 4.20 evaluates the reasons for non-use, using four factor loadings (F1, F2, F3 and F4). F1 comprises items that reflect uncertainty of rewards and complexity of use of SDMs. F2 is characterized with items that refer to lack of knowledge within and organisation. F3 consists of items patterning to cost accumulation and expertise availability. F4 categorizes items concerning compatibility with older systems and resistance to change.

The reliability of the factors was measured by means of Cronbach's alpha (CA). F1 and F2's Cronbach's alphas measured 0.84 and 0.82 respectively. Given that both CAs are above the industry recommended value of 0.70, it reflects that the reliabilities are good, thus proof that the validity of the variables is good. While F3 and F4 have CAs of 0.56 and 0.53 respectively, these values are below 0.70, thus the reliability can be said to not be good. Therefore, instead of treating the components of F3 and F4 as members of two factors, they can be treated as individuals

Table 4.20: Reasons for non-SDMs use factor analysis and reliability

Reasons for non-SDMs use				
Factor & Cronbach's alpha	Factor Name	Process factors	Questionnaire variables	Factor loading
F1: 0.84	Uncertainty of benefits and complexity of SDMs	In our IS department there are no clear objectives for adopting systems development methodologies	SBQ8.13	0.81
		Systems development methodologies are too complex or hard to use	SBQ8.2	0.80
		The financial investment in systems development methodologies is too large	SBQ8.11	0.79
		In our IS department there is a lot of uncertainty over the benefits of adopting systems development methodologies	SBQ8.12	0.76
		Profile of development projects in our IS department doesn't require the use of systems development methodologies	SBQ8.1	0.61

F2: 0.82	Knowledge deficiency	In our IS department there is a lack of management support for the use of systems development methodologies	SBQ8.9	0.91
		There is a lack of experienced staff in our IS department who can effectively use systems development methodologies.	SBQ8.6	0.82
		Our IS department lacks a suitable environment to support systems development methodologies	SBQ8.8	0.82
F3: 0.56	Direct cost and expertise availability	The experience of the developers in our IS department reduces the need for systems development methodologies	SBQ8.4	0.86
		The benefits of systems development methodology use are long-term, whereas costs are incurred short term	SBQ8.5	0.60
F4: 0.53	Incompatibility and contentment	The current systems development practice in our IS department is adequate	SBQ8.3	0.74
		New systems developed with systems development methodologies are not compatible with legacy systems	SBQ8.7	0.63
		The learning curve for systems development methodologies is very long.	SBQ8.10	0.61

4.5 SECTION C: Software Process Improvement Models Certification (Research objective 2)

This section is aimed at determining whether an organisation has been accredited to an industry model of standardisation in the form of Software Process Improvement Models (SPIMs). It aims to capture the types of - and the levels to which - SPMs are used within the organisation, as well as descriptions of development procedures and processes for certified organisations.

4.5.1 Software Process Improvement Models (SPIMs) usage

Seven respondents indicated that they were certified with CMMi, while 12 had ISO 900-3 certification. Only 2% were certified with SPICE, there rest of the respondents (62.8%) either did not know or specified that the organisation was not certified. The results of these findings are tabulated in Table 4.21 below.

Table 4.21: Software Process Improvement Models Certification

Frequency Table: SPIMs certification			
SPIM		**Frequency**	**Valid Percent**
Valid	CMMi	7	7.1
	ISO 9000-3	12	12.1
	SPICE	2	2
	No Certification	36	18.8
	Don’t know	44	44.0
	Total	101	84

4.5.2 CMMi level of certification

Six out of the seven respondents with CMMi certification also indicated the maturity level of CMMi. Level 2 and 3 each had two respondents. Similarly, the other two respondents said they were at level 1 and 5 respectively. Table 4.22 shows the results of maturity levels.

Table 4.22: CMMi certification level

Frequency Table: CMMi certification level					
Certification Level		**Frequency**	**Percent**	**Valid Percent**	**Cumulative Percent**
Valid	Level 1	1	.5	16.7	16.7
	Level 2	2	1.0	33.3	50.0
	Level 3	2	1.0	33.3	83.3
	Level 5	1	.5	16.7	100.0
	Total	6	3.1	100.0	
Missing	System	186	96.9		
Total		192	100.0		

4.5.3 Reason for certification

This question intended to capture the main factor that influenced organisations to be certified. These factors' influence and results are discussed and shown in Table 4.23 below.

- *Better market impression*
  Fifteen percent of the respondents agreed that the motivating factor for getting certification was to make a better impression on the market.

- *Better quality products*
  Market impression and production of quality products contributed equally (15%) to motivating organisations to become certified.

- *Faster development times*
  "Timeliness development of products" had 6% of respondents agreeing with the statement that quicker development time was a motivation for certification.

- *The required skills/tools are already available*
  Only 4% of the respondents agreed that the required skills or tools were already available, and thus a good motivation for certification.

Table 4.23: Reasons for certification

Frequency Table: Reasons for certification			
		Frequency	Valid Percent
Valid	Better market impression	15	15.0
	Better quality products	15	15.0
	Faster development times	6	6.0
	The required skills/tools are already available	4	4.0

4.5.4 System development methodology role on certification

This question was aimed at determining whether the SDM within an organisation played a role in what type of certification the organisation adopted. Seventeen of 19 respondents (89.5%) disagreed with the notion that the SDM in use had an influence on the type of certification the organisation would have, while only two of the respondents agreed - signifying that very few organisations got certified with a SPIM based on the type of SDM in use. The results of the question are displayed in Table 4.24 below.

Table 4.24: Influence of SDMs on certification type

Frequency Table: SDM role on type of certification					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NO	17	8.9	89.5	89.5
	YES	2	1.0	10.5	100.0
	Total	19	9.9	100.0	
Missing	System	173	90.1		
Total		192	100.0		

4.5.5 SPI Procedures and Processes

The statements in this section of the questionnaire represented software process improvement procedures and processes. These procedures and processes are activities that ought to be performed by an organisation, based on what maturity level the organisation is classified under. Table 4.25 shows the frequency table of these activities. In-depth analysis of these procedures and processes is done in the following

sub-sections.

Table 4.25: SPI procedures and processes

Frequency Table: Development procedures and processes						
Development processes and procedures	**YES**		**No**		**Totals**	
	Freq	**%**	**Freq**	**%**	**Total Freq**	**Total %**
Is a formal procedure used in the management review of each software development project prior to making contractual commitments?	60	71.4	24	28.6	84	100
Is a formal procedure used to make estimates of software size?	60	71.4	24	28.6	84	100
Is a formal procedure used to produce software development schedules?	70	83.3	14	16.7	84	100
Is a formal procedure used to make estimates of software development cost?	72	85.7	12	14.3	84	100
Do software development first-line managers sign off on their schedules and cost estimates?	57	67.9	27	32.1	84	100
Does senior management have a mechanism for the regular review of the status of software development projects?	71	83.5	14	16.5	85	100
Is there a software configuration control function for each project that involves software development?	52	63.4	30	36.6	82	100
Are profiles of software size maintained for each configuration items, over time?	42	52.5	38	47.5	80	100
Is a mechanism used for controlling changes to the software requirements? (Who can make changes and under what circumstances?)	68	80.0	17	20.0	85	100

Is a mechanism used for controlling changes to the software design? (Who can make changes and under what circumstances?)	69	84.1	13	15.9	82	100
Is a mechanism used for controlling changes to the code? (Who can make changes and under what circumstances?)	68	82.9	14	17.1	82	100
Is there a software engineering process group function?	30	38.5	48	61.5	78	100
Does your IS department use a standardized software development process?	60	75.0	20	25.0	80	100
Does your IS department use a standardized and documented software development process on each project?	51	63.0	30	37.0	81	100
Is a mechanism used for ensuring compliance with the software engineering standards?	40	51.9	37	48.1	77	100
Is there a required software engineering program for software developers?	36	46.8	41	53.2	77	100
Is a formal training program required for design and code review leaders?	35	45.5	42	54.5	77	100
Does the Software Quality Assurance (SQA) function have a management reporting channel separate from the software development project management?	33	42.9	44	57.1	77	100
Are internal software design reviews conducted?	59	72.0	23	28.0	82	100
Are the action items resulting from design reviews tracked to closure?	44	55.0	36	45.0	80	100
Are statistics on software design errors gathered?	33	42.3	45	57.7	78	100
Are software code reviews conducted?	55	68.8	25	31.3	80	100
Are the action items resulting from code reviews tracked to closure?	38	50.7	37	49.3	75	100
Are statistics on software code and test errors gathered?	38	48.1	41	51.9	79	100

Are design errors projected and compared to actual?	37	46.8	42	53.2	79	100
Are the review data, gathered during design reviews, analysed?	39	50.0	39	50.0	78	100
Are code and test errors projected and compared to actual?	37	48.1	40	51.9	77	100
Are the error data from code reviews and tests analysed, to determine the likely distribution and characteristics of errors remaining in the product?	31	41.3	44	58.7	75	100
Are design and code review coverage measured and recorded?	33	41.8	46	58.2	79	100
Is review efficiency analysed for each project?	33	40.7	48	59.3	81	100
Are code review standards applied?	48	59.3	33	40.7	81	100
Is test coverage measured and recorded for each phase of functional testing?	45	57.7	33	42.3	78	100
Is there a mechanism for assuring the adequacy of regression testing?	43	53.1	38	46.9	81	100
Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?	40	51.3	38	48.7	78	100
Has a managed and controlled process database been established for processing metrics data across all projects?	26	32.9	53	67.1	79	100
Is a mechanism used for periodically assessing the software engineering process, and implementing indicated improvements?	42	52.5	38	47.5	80	100
Are analyses of errors conducted to determine their process-related causes?	44	55.7	35	44.3	79	100
Is a mechanism used for managing and supporting the introduction of new technologies?	46	58.2	33	41.8	79	100

4.5.6 Descriptive statistics of processes and procedures

The processes and procedures are classified into three different classes or levels (level2, level3, and level4). Each process or procedure is an activity that should ideally be performed at a certain level. To compute the mean and standard deviation, the following formula in Figure 4.1 was used.

```
COMPUTE Level2=mean(statistics of all level 2 define activities).
EXECUTE.
COMPUTE Level3=mean(statistics of all level 3 define activities).
EXECUTE.
COMPUTE Level4=mean(statistics of all level 4 define activities).
EXECUTE.
```

Figure 4.2: CMMi Algorithm for computing level Mean.

Table 4.26 below presents the results of applying the CMMi algorithm to compute the mean and standard deviation for each class or level.

Table 4.26: mean and standard deviation

	N	Minimum	Maximum	Mean	Std. Deviation
Level2	85	0.00	1.00	.6997	.26155
Level3	84	0.00	1.00	.5794	.30048
Level4	82	0.00	1.00	.4895	.34445
Valid N (listwise)	82				

4.5.7 Maturity level classification Algorithm

An algorithm (Figure 4.2) has been developed to determine/classify the maturity level an organisation falls under, based on the processes and procedures performed. Below is a snippet of the algorithm. The class level is initialised to 1; all organisations are assumed to be at a maturity level of 1. If the mean of all activities that should be done at level 2 is greater or equal to 0.8, then the organisation will be classified at a maturity level of 2. Similarly, level 3 and level 4 are computed. If the mean of level 2 is greater or equal to 0.8 **and** mean of level 3 is greater or equal to 0.8, then the organisation will be classified at a maturity level of 3. Similarly, level 4's classification is computed.

```
COMPUTE Clas_level=1.
EXECUTE.
IF  (Level2>=0.8) Clas_level=2.
EXECUTE.
IF  (Level2>=0.8 and level3>=0.8) Clas_level=3.
EXECUTE.
IF  (Level2>=0.8 and level3>=0.8 and level4>=0.8) Clas_level=4.
EXECUTE.
```

Figure 4.3: Algorithm for maturity level classification.

The results in Table 4.27 show the computed maturity levels of the respondents. A total of 79.2% were classified at maturity level 1; 8.9% at level 2; 3.6% at level 3; and 8.3% were involved in activities that would classify them at maturity level 4.

Table 4.27: Maturity level classification

Organisation class levels					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Level_1	152	79.2	79.2	79.2
	Level_2	17	8.9	8.9	88.0
	Level_3	7	3.6	3.6	91.7
	Level_4	16	8.3	8.3	100.0
	Total	192	100.0	100.0	

4.5.8 Activity distribution per maturity level

Each process and procedure falls under a generally agreed class level. Table 4.28 presents these processes and procedures with the respective class levels they fall under. Crosstabs were computed to determine the percentages of these activities performed at each maturity level.

A common occurrence seen in the results obtained is that activities were performed haphazardly. For instance, an activity such as having a mechanism for controlling changes to the software design, is an activity that is performed at class level 3. Across the board, the results show that 100% at level 3 did perform this function. However, the results also indicate that 73.3% and 93.8% (representing level 1 and level 2 respectively) performed the activity (which is above their classification). Thus we must reach the conclusion that the practices in the industry do not follow the stipulated standards.

Table 4.28: Activities performed at each class level

Percentage of activities performed at each class level									
		Class_level 1		Class_level 2		Class_level 3		Class_level 4	
Activity Class Level	Procedures & Processes	NO	YES	NO	YES	NO	YES	NO	YES
2	Is a formal procedure used in the management review of each software development project prior to making contractual commitments?	50.0%	50.0%	0.0%	100.0%	0.0%	100.0%	12.5%	87.5%
2	Is a formal procedure used to make estimates of software size?	54.5%	45.5%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%
2	Is a formal procedure used to produce software development schedules?	27.3%	72.7%	0.0%	100.0%	0.0%	100.0%	12.5%	87.5%
2	Is a formal procedure used to make estimates of software development cost?	27.3%	72.7%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%
2	Do software development first-line managers sign off on their schedules and cost estimates?	54.5%	45.5%	5.9%	94.1%	0.0%	100.0%	12.5%	87.5%
2	Does senior management have a mechanism for the regular review of the status of software development projects?	31.1%	68.9%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%
2	Is there a software configuration control function for each project that involves software development?	59.5%	40.5%	17.6%	82.4%	14.3%	85.7%	6.3%	93.8%

2	Are profiles of software size maintained for each configuration items, over time?	73.8%	26.2%	23.5%	76.5%	14.3%	85.7%	14.3%	85.7%
2	Is a mechanism used for controlling changes to the software requirements? (Who can make changes and under what circumstances?)	33.3%	66.7%	5.9%	94.1%	0.0%	100.0%	6.3%	93.8%
3	Is a mechanism used for controlling changes to the software design? (Who can make changes and under what circumstances?)	26.7%	73.3%	6.3%	93.8%	0.0%	100.0%	0.0%	100.0%
2	Is a mechanism used for controlling changes to the code? (Who can make changes and under what circumstances?)	31.1%	68.9%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%
3	Is there a software engineering process group function?	78.6%	21.4%	75.0%	25.0%	16.7%	83.3%	14.3%	85.7%
3	Does your IS department use a standardized software development process?	34.1%	65.9%	26.7%	73.3%	0.0%	100.0%	7.1%	92.9%
3	Does your IS department use a standardized and documented software development process on each project?	58.1%	41.9%	31.3%	68.8%	0.0%	100.0%	0.0%	100.0%
3	Is a mechanism used for ensuring compliance with the software engineering standards?	65.9%	34.1%	50.0%	50.0%	0.0%	100.0%	14.3%	85.7%
3	Is there a required software engineering program for software developers?	70.7%	29.3%	50.0%	50.0%	33.3%	66.7%	14.3%	85.7%

3	Is a formal training program required for design and code review leaders?	69.0%	31.0%	66.7%	33.3%	33.3%	66.7%	7.1%	92.9%
2	Does the Software Quality Assurance (SQA) function have a management reporting channel separate from the software development project management?	81.4%	18.6%	42.9%	57.1%	33.3%	66.7%	7.1%	92.9%
3	Are internal software design reviews conducted?	45.5%	54.5%	18.8%	81.3%	0.0%	100.0%	0.0%	100.0%
3	Are the action items resulting from design reviews tracked to closure?	72.1%	27.9%	33.3%	66.7%	0.0%	100.0%	0.0%	100.0%
3	Are statistics on software-design errors gathered?	95.2%	4.8%	25.0%	75.0%	16.7%	83.3%	0.0%	100.0%
3	Are software code reviews conducted?	47.7%	52.3%	25.0%	75.0%	0.0%	100.0%	0.0%	100.0%
3	Are the action items resulting from code reviews tracked to closure?	75.0%	25.0%	46.7%	53.3%	0.0%	100.0%	0.0%	100.0%
2	Are statistics on software code and test errors gathered?	86.0%	14.0%	18.8%	81.3%	16.7%	83.3%	0.0%	100.0%
4	Are design errors projected and compared to actual?	79.1%	20.9%	31.3%	68.8%	33.3%	66.7%	7.1%	92.9%
4	Are the review data, gathered during design reviews, analysed?	74.4%	25.6%	25.0%	75.0%	40.0%	60.0%	7.1%	92.9%
4	Are code and test errors projected and compared to actual?	76.2%	23.8%	42.9%	57.1%	16.7%	83.3%	6.7%	93.3%

4	Are the error data from code reviews and tests analysed, to determine the likely distribution and characteristics of errors remaining in the product?	78.0%	22.0%	53.3%	46.7%	50.0%	50.0%	7.7%	92.3%
4	Are design and code review coverage measured and recorded?	79.1%	20.9%	53.3%	46.7%	66.7%	33.3%	0.0%	100.0%
4	Is review efficiency analysed for each project?	76.7%	23.3%	56.3%	43.8%	100.0%	0.0%	0.0%	100.0%
4	Are code review standards applied?	61.4%	38.6%	13.3%	86.7%	50.0%	50.0%	6.3%	93.8%
4	Is test coverage measured and recorded for each phase of functional testing?	61.0%	39.0%	40.0%	60.0%	33.3%	66.7%	0.0%	100.0%
3	Is there a mechanism for assuring the adequacy of regression testing?	76.7%	23.3%	25.0%	75.0%	16.7%	83.3%	0.0%	100.0%
3	Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?	76.2%	23.8%	25.0%	75.0%	33.3%	66.7%	0.0%	100.0%
4	Has a managed and controlled process database been established for process metrics data across all projects?	90.7%	9.3%	62.5%	37.5%	66.7%	33.3%	0.0%	100.0%
4	Is a mechanism used for periodically assessing the software engineering process, and implementing indicated improvements?	76.2%	23.8%	31.3%	68.8%	16.7%	83.3%	0.0%	100.0%
4	Are analyses of errors conducted to determine their process-related causes?	65.1%	34.9%	31.3%	68.8%	33.3%	66.7%	0.0%	100.0%

4	Is a mechanism used for managing and supporting the introduction of new technologies?	56.1%	43.9%	50.0%	50.0%	16.7%	83.3%	6.3%	93.8%

4.5.9 Reasons for non-SPIM certification

The statements in this section assessed the contributing factors as to why the organisation was not certified. The extent to which each statement contributes to non-certification is measured. The results of this question are discussed below and reported in table 4.29.

- *Does not meet required criteria*
  A total of 2.6% of the respondents stated that their organisation was not SPIM-certified because the organisation did not meet the required criteria.
- *There are no benefits or advantages*
  Most of those who responded (10.4%) thought that SPIMs have neither benefits nor advantages, thus there is no need to be certified.
- *Lack of funds*
  A total of 4.7 of the respondents pinned the reason for no certification on the unavailability of funding.
- *Lack of benefits and advantages as well as lack of skills and tools*
  One respondent stated that two factors contributed to the organisation not being certified, viz, lack of benefit and advantages, and lack of skills and tools.

Table 4.29: Reason s for non-certification

Frequency Table: Non certification factors					
Factors for non-certification		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		140	72.9	72.9	72.9
	Does not meet required criteria	5	2.6	2.6	75.5
	There are no benefits or advantages	20	10.4	10.4	85.9
	Lack of benefits and advantages as well as lack of skills and tools	1	0.5	0.5	86.5
	Lack of funds	9	4.7	4.7	91.1
	The required skills/tools are not available	7	3.6	3.6	94.8
	Other	10	5.2	5.2	100.0
	Total	192	100.0	100.0	

4.6 SECTION D: The project outcome (Objective 3)

This section describes the outcome of IS projects in which the organisation or respondents were involved, and contributing factors that aided the successful completion of the project.

4.6.1 Last project outcome

The statements in this question described the status of the last project with which the respondents were involved. The findings of each statement are discussed and shown below in Table 4.30.

- *The project was cancelled/terminated before completion.*
  Of the 98 respondents, only six respondents said that the project was cancelled or terminated before completion.
- *The project was completed but not implemented.*
  Four projects had been completed but were not implemented.
- *The project was completed and implemented, but is not in use anymore.*
  Only one completed and implemented project is no longer in use today.
- *The project was completed and implemented, and is still in use.*
  A significant 86 of 95 respondents (87.8%) indicated that the project was completed and implemented and is currently in use.

Table 4.30: Statements describing project outcome

Frequency Table: Last project outcome					
Project outcome		**Frequency**	**Percent**	**Valid Percent**	**Cumulative Percent**
Valid	The project was cancelled/terminated before time.	6	3.1	6.1	6.1
	The project was completed but not implemented.	4	2.1	4.1	10.2
	The project was completed and implemented, but is not in use anymore.	1	.5	1.0	11.2
	The project was completed and implemented, and is still in use.	86	44.8	87.8	99.0
	Other reason	1	.5	1.0	100.0
	Total	98	51.0	100.0	
Missing	System	94	49.0		
Total		192	100.0		

4.6.2 Life-span of last project involved in

Only 38.2 percent of completed projects had been in use for over one year. The longest period that a complete and implemented project had been in use is 120 months. The rest had been in use only for a matter of months, and nearly 15% of these projects had only been rolled out for six months. Table 4.31 indicates the life span of the last system development methodologies respondents were involved in.

Table 4.31: Life-span of SDMs

Frequency Table: Life-span of SDMs					
Period		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1 Month	5	2.6	6.8	6.8
	2 Months	3	1.6	4.1	10.8
	3 Months	7	3.6	9.5	20.3
	4 Months	6	3.1	8.1	28.4
	5 Months	3	1.6	4.1	32.4
	6 Months	11	5.7	14.9	47.3
	7 Months	3	1.6	4.1	51.4
	8 Months	2	1.0	2.7	54.1
	9 Months	3	1.6	4.1	58.1
	10 Months	3	1.6	4.1	62.2
	12 Months	6	3.1	8.1	70.3
	14 Months	3	1.6	4.1	74.3
	16 Months	2	1.0	2.7	77.0
	18 Months	3	1.6	4.1	81.1
	20 Months	1	.5	1.4	82.4
	24 Months	6	3.1	8.1	90.5
	36 Months	2	1.0	2.7	93.2
	48 Month	1	.5	1.4	94.6
	60 Month	1	.5	1.4	95.9
	72 Months	1	.5	1.4	97.3
	96 Months	1	.5	1.4	98.6
	120 Months	1	.5	1.4	100.0
	Total	74	38.5	100.0	
Missing	System	118	61.5		
Total		192	100.0		

4.6.3 Project success: process factors influencing project success

This section and the next explore factors that contributed to successful completion of a project, from a process perspective as well as the actual end product. The statements in this question measured the contribution of a number of factors in the successful completion of the last projects in which the respondents were involved. These factors aimed to illustrate that project-management success is measured by the ability to

manage the knowledge areas of project management, especially those that form part of the triple constraint. That is, the ability to manage the cost, scope and schedule of the project whilst maintaining high quality standards of the end product. Table 4.32 shows the summary of the frequency distribution of all the respondents in this question. The results of each factor are discussed separately.

- *The project was completed on schedule*
  A total of 45.7% of the respondents generally agreed to the statement and 36.1% of the respondents disagreed. Thus we can conclude that last projects tackled were generally completed timeously, and this is a positive result.
- *The project was completed within budget*
  Exactly 52.5% of the respondents (43 out of 82) generally agreed to this statement and only 16 other respondents (19.57%) disagreed. The rest remained neutral. Therefore, slightly more than half of the projects finished on budget.
- *The developed system satisfied all the stated requirements*
  An overwhelming majority of 72.9% of the respondents agreed with the statement, and only 4.9% disagreed. The remaining 22.2% of the respondents were indifferent to the statement. Therefore it can be concluded that the last systems developed satisfied all the stated requirements.
- *The speed of developing the project was high*
  A total of 59.2% of the respondents agreed with the statement and almost 10% of respondents disagreed. The remaining 31% of respondents were neutral to the statement. Therefore, this result shows that generally the speed of development was high in the last SD project.
- *The productivity of developers involved with the project was high*
  A large percentage of the respondents (64.6%) agreed to this statement, suggesting that generally productivity levels of the developers were fairly high.
- *The cost of the project was low when compared to the size and complexity of the system developed*
  Exactly 50% of the respondents agreed with this statement and 10 respondents (12.2%) disagreed. The remaining respondents (37.8%) were neutral to this statement. Therefore it can be said that even though half of the respondents felt that the cost was low with regards to the size and complexity of the system, a large

contingent was sitting on the fence.

- *The project achieved its goals*
  Overwhelmingly, 84% of the respondents generally agreed to this statement. Only one respondent disagreed and the remaining 14.8% were neutral to the statement. Therefore this result shows that the last project generally achieved its goals.
- *Overall, the project represents excellent work*
  A total of 64.2% of the respondents generally agreed with this statement. Therefore these results show that the majority of the respondents feel that the project they were involved with represented excellent work. Meanwhile, 25% of the respondents were undecided.
- *Overall, the project was a success*
  A large proportion, 84.1% of the respondents, generally agreed to this statement, signifying that the last project that the respondents were involved with was largely a success.

This question's results generally yielded positive outcomes in all the statements above. Therefore, it can be concluded that the last projects that the respondents were involved with were completed within the allocated budget and time. Furthermore, the productivity of the developers was considered to be of a considerably high standard, and the costs involved were low compared to the size and complexity of the project. Thus it can be said that in general the project was a success and it represented good work. Table 4.26 shows a summary of results of the frequency distribution.

Table 4.32: Summary of process factors affecting project success rate

Frequency Table: Process factors that Influence the project success												
Factors affecting project success	**Totally Disagree**		**Disagree**		**Neutral**		**Agree**		**Totally Agree**			
	Freq	**%**	**Freq**	**%**	**Freq**	**%**	**Freq**	**%**	**Freq**	**%**	**Total Freq**	**Mean**
The project was completed on schedule.	5	6.2	21	25.9	18	22.2	25	30.9	12	14.8	81	3.22
The project was completed within budget.	1	1.2	15	18.3	23	28.0	25	30.5	18	22.0	82	3.54
The developed system satisfied all the stated requirements.	1	1.2	3	3.7	18	22.2	43	53.1	16	19.8	81	3.86
Speed of developing new applications was high.	0	0	8	9.9	25	30.9	38	46.9	10	12.3	81	3.62
The productivity involved in the projects was high.	0	0	4	5.1	24	30.4	39	49.4	12	15.2	79	3.75
The cost of project is low when compared to the size and complexity of the system developed.	2	2.4	8	9.8	31	37. 8	27	32.9	14	17.1	82	3.52
The project achieved its goal.	0	0	1	1.2	12	14.8	43	53.1	25	30.9	81	4.14
Overall, the project represents excellent work.	2	2.5	7	8.6	20	24.7	34	42.0	18	22.2	81	3.73
Overall, project was a success.	1	1.2	1	1.2	11	13.4	43	52.4	26	31.7	82	4.12

4.6.3.1 Factor analysis and reliability

The various process success variables are grouped into two un-rotated factor loadings to determine the nature of the construct influencing observed process success variables. Table 4.33 measures the outcome of the project using two factor loadings (F1 and F2). F1 comprises of variables that reflect the scope, time and cost management processes, so it is named “Process triple constraint management”. F2 comprises variables that illustrate the goal achievement process, so it is referred to as “process objectives and goals”. The reliability of the factors was measured by means of Cronbach’s alpha (CA). F1 and F2’s Cronbach’s alphas measured 0.77 and 0.82 respectively. Given that both CA’s are above the industry recommended value of 0.70, it reflects that the reliabilities are good, thus proof that the validity of the variables is good.

Table 4.33: Project success factor analysis and reliability

Project success factor analysis and reliability				
Factor & Cronbach’s alpha	Factor Name	Process influencing factors	Questionnaire variables	Factor loading
F1: 0.77	Process triple constraint management	The project was completed on schedule.	SDQ3.1	0.82
		Speed of developing new applications was high.	SDQ3.4	0.82
		The project was completed within budget.	SDQ3.2	0.76
		The productivity involved in the projects was high	SDQ3.5	0.66
		Cost of project is low when compared to size and complexity of system developed	SDQ3.6	0.52
F2: 0.82	Process objectives and goals	The project achieved its goal	SDQ3.7	0.89
		Overall, the project represents excellent work	SDQ3.8	0.82
		Overall, the project was success	SDQ3.9	0.80
		The developed system satisfied all the stated requirements	SDQ3.3	0.72

4.6.4 Project success: End-product factors influencing project success

The statements in this question measured the scope of the actual system that was developed in terms of functionality, maintainability, reliability, and efficiency. These statements aim to measure the overall quality of the last system developed. Similarly to 4.6.3, each statement is discussed separately. Table 4.34 shows results of the summary of the project's success in terms of the systems scope.

- *The functionality of the developed system is high*

A total of 83.5% of the respondents generally agreed with this statement. This overwhelmingly high percentage signifies that the respondents viewed the functionality of the system as being high. Only 4.8% (four out of 85) respondents disagreed with the statement.

- *The reliability of the developed system is high*

Reliability of the system had 78.8% of the respondents (67 out of 85) generally agreeing with the statement. Thus it can be said that the reliability of the last systems the respondents developed was of high standards.

- *The maintainability of the developed system is high*

Close to 70% of the respondents generally agreed with the statement that the maintainability of the developed systems was high.

- *The portability of the developed system is high.*

A total 51.8% of respondents generally agreed with the statement, though not as overwhelmingly as with other statements. Over 30% of the respondents chose to remain neutral. The positive result signifies that generally the portability of the developed system is high.

- *The efficiency of the developed system is high*

The efficiency of the developed system had 71.5% of the respondents agreeing with the statement. This high percentage of positive responses generally signifies that there was a consensus that the system was highly efficient.

- *The usability of the developed system is high.*

A total of 81.2% of the respondents (69 out of 85) generally agreed with the statement. This result illustrates that generally the usability of the developed system was high.

- *The developed system meets user needs*

Most of the respondents (84.5%) agreed with this statement. This signifies that the respondents felt that the system met the needs of the users.

- *The documentation of the developed system is good*

A total of 30.6% of the respondents generally disagreed with the statement. Although 42.4% generally felt that the documentation is good, this result is not as positive, and signifies that the documentation of the last project's development was not so good.

- *Overall, the quality of the developed system is high*

  Nearly 84% of the respondents (71 out of 85) generally agreed with the statement. This result shows that the quality of the developed system was high.

- *Overall, the users are satisfied with the developed system*

A total of 76.5% of the respondents agreed with the statement, meaning that overall users are satisfied with the developed system.

- *Overall, the developed system is a success*

More than 81% of the respondents agreed to this statement, meaning that the last developed system was considered to be successful. The frequency mean of 4.02 also indicates that the respondents generally agree. All the statements from this section yielded high positive responses, showing that the respondents felt that the last developed system was of a high quality.

Table 4.34: Summary on overall System functionality

System scope factors	Totally Disagree		Disagree		Neutral		Agree		Totally Agree		Totals	
	Freq	%	Freq	%	Freq	%	Freq	%	Freq	%	Total Freq	Freq Mean
The functionality of the developed system is high.	2	2.4	2	2.4	10	11.8	49	57.6	22	25.9	85	4.02
The reliability of the developed system is high.			5	5.9	13	15.3	46	54.1	21	24.7	85	3.98
The maintainability of the developed system is high.			8	9.6	18	21.7	45	54.2	12	14.5	83	3.73
The portability of the developed system is high.	2	2.4	12	14.1	27	31.8	36	42.4	8	9.4	85	3.42
The efficiency of the developed system is high.			3	3.6	21	25.0	45	53.6	15	17.9	84	3.86
The usability of the developed system is high.			6	7.1	10	11.8	50	58.8	19	22.4	85	3.96
The developed system meets users' needs.			2	2.4	11	13.1	48	57.1	23	27.4	84	4.10
The documentation of the developed system is good.	6	7.1	20	23.5	23	27.1	26	30.6	10	11.8	85	3.16
Overall, the quality of the developed system is high.	1	1.2	4	4.7	9	10.6	58	68.2	13	15.3	85	3.92
Overall, the users are satisfied with the developed system.			5	5.9	15	17.6	48	56.5	17	20.0	85	3.91
Overall, developed system is a success			2	2.4	14	16.5	49	57.6	20	23.5	85	4.02

4.6.4.1 Factor analysis and reliability

Similarly to process-success factors, variables are grouped into two un-rotated factor loadings to determine the nature of the construct influencing observed system scope success variables. Table 4.35 measures the outcome of the project using two factor loadings (F1 and F2). F1 comprises variables that are aligned to the effectiveness of the system, and thus an indication of whether the intended goal has been achieved. These variables can be grouped under one factor category as “System and goal success”. F2 comprises variables that play a supportive role to the system once it has been rolled out, such as system documentation, portability and maintainability, thus we categorize them as “system support”. Once more, Cronbach’s alpha was measured to determine the reliability of the factors. F1 and F2’s Cronbach’s alphas measured 0.92 and 0.71 respectively. Both CAs indicate that the reliabilities are good, thus proof that the validity of the variables is good.

Table 4.35: System outcome functionality analysis and reliability

Project success factor analysis and reliability				
Factor & Cronbach alpha	Factor Name	Product influencing factors	Questionnaire variables	Factor loading
F1: 0.92	System and goal success	The functionality of the developed system is high	SDQ4.1	0.82263
		The reliability of the developed system is high.	SDQ4.2	0.79116
		The usability of the developed system is high	SDQ4.6	0.51781
		The developed system meets users’ needs	SDQ4.7	0.90217
		Overall, the quality of the developed system is high	SDQ4.9	0.75426
		Overall, the users are satisfied with the developed system.	SDQ4.10	0.87199
		Overall, the developed system is a success	SDQ4.11	0.84243
F2:0.71	System support	The maintainability of the developed system is high	SDQ4.3	0.397
		The portability of the developed system is high	SDQ4.4	0.78454
		The efficiency of the developed system is high	SDQ4.5	0.59896
		The documentation of developed system is good	SDQ4.8	0.84477

4.6.5 Statistical analysis: Components of factor analysis for project success

The factors contributing to project success, that is those from processes and procedures followed in project development, all have high mean values (above 3), an indication that the factors are all valuable. Table 4.36 presents the four factors in descending order of their means.

Table 4.36: The statistical analysis of the project success factors

Success factor	Valid N	Mean	Minimum	Maximum	Std. Deviation
System and goal success	85.00	3.99	2.00	5.00	0.62
Process objectives and goals	82.00	3.97	2.00	5.00	0.68
System support	85.00	3.54	1.75	5.00	0.68
Process triple constraint management	82.00	3.53	2.00	5.00	0.70

To summarize section D of the research questionnaire, it quantified the outcome of the projects with which the respondents had been involved. It evaluated the process factors that influenced the outcome of the project, as well as the factors that influence effectiveness of the completed product.

4.7 Combination of SDMs and SPIMs use in organisation

This section aims to present the results of how much a system development methodology is used when a certain software process improvement model is in place or not. Crosstabs performed per methodology against each model yielded the following results, displayed in Table 4.37 below.

The Table of results shows each methodology and process model combination and the effect size at each extent of use. For instance, the combination of SDLC and CMMi reveals that when the methodology is nominally used, companies do not use CMMi at all.

In general, majority of the system development methodologies combined well with ISO-900-3. However, in most instances, the absence of a process model (non-certification) resulted in most respondents indicating that they did use that particular methodology. The highest combination yielded was when the respondents were not aware of the presence of process improvement model within the organisation.

The effect size or practical significance of crosstabs between the extent of SDM use per SPIMs is represented by phi coefficient or Cramer's V (Steyn, 2002). Steyn (2002) gives the following guidelines for judging the importance of a relationship:

- If phi is~0.1 then there is a practical non-significant association between the variables
- If phi is ~0.3, then the relationship is a practical visible significant association and,
- If phi is~0.5, then the association is practically significant or large effect.

Table 4.37 shows the phi values for each relationship between ASDMs and SPIMs. If we pay attention to SDLC, we notice that the relationship with other SPIMs is mostly a non-significant association or it has a small effect

Table 4.37: Extent of methodology use per software improvement model

	Extent of use of each methodology per SPIM													
	Nominal		**Mild**		**Average**		**Extensive**		**Extensively**		**Effect Size**	**Total**		**Total**
SDM*SPIM	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes		No	Yes	
SDLC * CMMi	100.00 %	0.00%	87.50%	12.50 %	94.10%	5.90%	90.50%	9.50%	88.90%	11.10%	0.15	92.20%	7.80%	100.0 0%
SDLC * ISO 9000-3	92.90%	7.10%	87.50%	12.50 %	82.40%	17.60 %	95.20%	4.80%	81.50%	18.50%	0.18	87.90%	12.10 %	100.0 0%
SDLC * SPICE	92.90%	7.10%	100.00 %	0.00%	100.00 %	0.00%	100.00 %	0.00%	96.30%	3.70%	0.18	97.80%	2.20%	100.0 0%
SDLC * No Certification	50.00%	50.00 %	62.50%	37.50 %	58.80%	41.20 %	76.20%	23.80 %	74.10%	25.90%	0.21	64.30%	35.70 %	100.0 0%
SDLC * Unaware	50.00%	50.00 %	62.50%	37.50 %	64.70%	35.30 %	52.40%	47.60 %	50.00%	50.00%	0.12	55.90%	44.10 %	100.0 0%
IE * CMMi	87.00%	13.00 %	83.30%	16.70 %	93.80%	6.30%	100.00 %	0.00%	100.00 %	0.00%	0.20	92.80%	7.20%	100.0 0%
IE * ISO 9000-3	91.30%	8.70%	75.00%	25.00 %	81.30%	18.80 %	77.80%	22.20 %	100.00 %	0.00%	0.21	85.10%	14.90 %	100.0 0%
IE * SPICE	100.00 %	0.00%	91.70%	8.30%	93.80%	6.30%	100.00 %	0.00%	100.00 %	0.00%	0.21	97.10%	2.90%	100.0 0%
IE * No Certification	56.50%	43.50 %	58.30%	41.70 %	75.00%	25.00 %	88.90%	11.10 %	75.00%	25.00%	0.25	70.70%	29.30 %	100.0 0%
IE * Unaware	65.20%	34.80 %	83.30%	16.70 %	43.80%	56.30 %	44.40%	55.60 %	50.00%	50.00%	0.30	57.30%	42.70 %	100.0 0%
RUP * CMMi	90.00%	10.00 %	85.70%	14.30 %	90.90%	9.10%	88.90%	11.10 %	0.00%	0.00%	0.06	71.10%	8.90%	80.00 %
RUP * ISO 9000-3	90.00%	10.00 %	71.40%	28.60 %	72.70%	27.30 %	100.00 %	0.00%	0.00%	0.00%	0.29	66.80%	13.20 %	80.00 %
RUP * SPICE	96.70%	3.30%	100.00 %	0.00%	90.90%	9.10%	100.00 %	0.00%	0.00%	0.00%	0.18	77.50%	2.50%	80.00 %
RUP * No Certification	56.70%	43.30 %	71.40%	28.60 %	81.80%	18.20 %	66.70%	33.30 %	0.00%	0.00%	0.20	55.30%	24.70 %	80.00 %
RUP * Unaware	56.70%	43.30 %	78.60%	21.40 %	45.50%	54.50 %	66.70%	33.30 %	0.00%	0.00%	0.23	49.50%	30.50 %	80.00 %
Scrum * CMMi	86.40%	13.60 %	81.30%	18.80 %	94.10%	5.90%	100.00 %	0.00%	100.00 %	0.00%	0.24	92.30%	7.70%	100.0 0%
Scrum * ISO	81.80%	18.20	93.80%	6.30%	88.20%	11.80	88.90%	11.10	90.00%	10.00%	0.13	88.50%	11.50	100.0

9000-3		%				%		%					%	0%
Scrum * SPICE	95.50%	4.50%	100.00 %	0.00%	94.10%	5.90%	100.00 %	0.00%	100.00 %	0.00%	0.16	97.90%	2.10%	100.0 0%
Scrum * No Certification	59.10%	40.90 %	81.30%	18.80 %	70.60%	29.40 %	44.40%	55.60 %	20.00%	80.00%	0.39	55.10%	44.90 %	100.0 0%
Scrum * Unaware	72.70%	27.30 %	56.30%	43.80 %	41.20%	58.80 %	50.00%	50.00 %	80.00%	20.00%	0.29	60.00%	40.00 %	100.0 0%
RAD * CMMi	92.90%	7.10%	85.70%	14.30 %	88.20%	11.80 %	94.40%	5.60%	90.90%	9.10%	0.11	90.40%	9.60%	100.0 0%
RAD * ISO 9000-3	92.90%	7.10%	78.60%	21.40 %	88.20%	11.80 %	83.30%	16.70 %	90.90%	9.10%	0.15	86.80%	13.20 %	100.0 0%
RAD * SPICE	92.90%	7.10%	100.00 %	0.00%	100.00 %	0.00%	94.40%	5.60%	100.00 %	0.00%	0.19	97.50%	2.50%	100.0 0%
RAD * No Certification	57.10%	42.90 %	64.30%	35.70 %	76.50%	23.50 %	55.60%	44.40 %	72.70%	27.30%	0.18	65.20%	34.80 %	100.0 0%
RAD * Unaware	57.10%	42.90 %	92.90%	7.10%	41.20%	58.80 %	61.10%	38.90 %	45.50%	54.50%	0.36	59.50%	40.50 %	100.0 0%
XP * CMMi	96.40%	3.60%	85.00%	15.00 %	92.90%	7.10%	85.70%	14.30 %	100.00 %	0.00%	0.20	92.00%	8.00%	100.0 0%
XP * ISO 9000-3	89.30%	10.70 %	80.00%	20.00 %	78.60%	21.40 %	100.00 %	0.00%	100.00 %	0.00%	0.22	89.60%	10.40 %	100.0 0%
XP * SPICE	96.40%	3.60%	100.00 %	0.00%	92.90%	7.10%	100.00 %	0.00%	100.00 %	0.00%	0.17	97.90%	2.10%	100.0 0%
XP * No Certification	53.60%	46.40 %	55.00%	45.00 %	71.40%	28.60 %	42.90%	57.10 %	100.00 %	0.00%	0.28	64.60%	35.40 %	100.0 0%
XP * Unaware	57.10%	42.90 %	75.00%	25.00 %	57.10%	42.90 %	71.40%	28.60 %	60.00%	40.00%	0.17	64.10%	35.90 %	100.0 0%
LSD * CMMi	94.30%	5.70%	75.00%	25.00 %	100.00 %	0.00%	66.70%	33.30 %	100.00 %	0.00%	0.34	87.20%	12.80 %	100.0 0%
LSD * ISO 9000-3	85.70%	14.30 %	83.30%	16.70 %	81.80%	18.20 %	66.70%	33.30 %	100.00 %	0.00%	0.14	83.50%	16.50 %	100.0 0%
LSD * SPICE	97.10%	2.90%	100.00 %	0.00%	100.00 %	0.00%	66.70%	33.30 %	100.00 %	0.00%	0.39	92.80%	7.20%	100.0 0%
LSD * No Certification	54.30%	45.70 %	83.30%	16.70 %	81.80%	18.20 %	100.00 %	0.00%	100.00 %	0.00%	0.35	83.90%	16.10 %	100.0 0%
LSD * Unaware	65.70%	34.30 %	50.00%	50.00 %	54.50%	45.50 %	33.30%	66.70 %	0.00%	100.00 %	0.27	40.70%	59.30 %	100.0 0%
FDD * CMMi	90.00%	10.00 %	71.40%	28.60 %	94.10%	5.90%	90.00%	10.00 %	100.00 %	0.00%	0.21	89.10%	10.90 %	100.0 0%

FDD * ISO 9000-3	86.70%	13.30 %	85.70%	14.30 %	82.40%	17.60 %	90.00%	10.00 %	100.00 %	0.00%	0.10	88.90%	11.10 %	100.0 0%
FDD * SPICE	96.70%	3.30%	100.00 %	0.00%	100.00 %	0.00%	90.00%	10.00 %	100.00 %	0.00%	0.19	97.30%	2.70%	100.0 0%
FDD * No Certification	53.30%	46.70 %	57.10%	42.90 %	70.60%	29.40 %	90.00%	10.00 %	100.00 %	0.00%	0.30	74.20%	25.80 %	100.0 0%
FDD * Unaware	66.70%	33.30 %	71.40%	28.60 %	52.90%	47.10 %	30.00%	70.00 %	50.00%	50.00%	0.27	54.20%	45.80 %	100.0 0%
Crystal * CMMi	87.80%	12.20 %	100.00 %	0.00%	87.50%	12.50 %	100.00 %	0.00%	100.00 %	0.00%	0.17	95.10%	4.90%	100.0 0%
Crystal * ISO 9000-3	87.80%	12.20 %	75.00%	25.00 %	75.00%	25.00 %	66.70%	33.30 %	66.70%	33.30%	0.20	74.20%	25.80 %	100.0 0%
Crystal * SPICE	97.60%	2.40%	100.00 %	0.00%	87.50%	12.50 %	100.00 %	0.00%	100.00 %	0.00%	0.21	97.00%	3.00%	100.0 0%
Crystal * No Certification	61.00%	39.00 %	75.00%	25.00 %	75.00%	25.00 %	66.70%	33.30 %	100.00 %	0.00%	0.20	75.50%	24.50 %	100.0 0%
Crystal * Unaware	56.10%	43.90 %	62.50%	37.50 %	62.50%	37.50 %	66.70%	33.30 %	66.70%	33.30%	0.08	62.90%	37.10 %	100.0 0%
DSDM * CMMi	89.20%	10.80 %	100.00 %	0.00%	91.70%	8.30%	100.00 %	0.00%	50.00%	50.00%	0.28	86.20%	13.80 %	100.0 0%
DSDM * ISO 9000-3	86.50%	13.50 %	100.00 %	0.00%	75.00%	25.00 %	66.70%	33.30 %	50.00%	50.00%	0.28	75.60%	24.40 %	100.0 0%
DSDM * SPICE	97.30%	2.70%	100.00 %	0.00%	91.70%	8.30%	100.00 %	0.00%	100.00 %	0.00%	0.15	97.80%	2.20%	100.0 0%
DSDM * No Certification	54.10%	45.90 %	87.50%	12.50 %	91.70%	8.30%	100.00 %	0.00%	100.00 %	0.00%	0.41	86.60%	13.40 %	100.0 0%
DSDM * Unaware	70.30%	29.70 %	12.50%	87.50 %	33.30%	66.70 %	33.30%	66.70 %	100.00 %	0.00%	0.47	49.90%	50.10 %	100.0 0%
ASD * CMMi	91.20%	8.80%	88.90%	11.10 %	88.90%	11.10 %	100.00 %	0.00%	100.00 %	0.00%	0.14	93.80%	6.20%	100.0 0%
ASD * ISO 9000-3	85.30%	14.70 %	88.90%	11.10 %	77.80%	22.20 %	80.00%	20.00 %	100.00 %	0.00%	0.12	86.40%	13.60 %	100.0 0%
ASD * SPICE	97.10%	2.90%	100.00 %	0.00%	88.90%	11.10 %	100.00 %	0.00%	100.00 %	0.00%	0.20	97.20%	2.80%	100.0 0%
ASD * No Certification	55.90%	44.10 %	55.60%	44.40 %	88.90%	11.10 %	70.00%	30.00 %	100.00 %	0.00%	0.28	74.10%	25.90 %	100.0 0%
ASD * Unaware	61.80%	38.20 %	77.80%	22.20 %	44.40%	55.60 %	50.00%	50.00 %	50.00%	50.00%	0.20	56.80%	43.20 %	100.0 0%

4.8 The Correlations between SDMs and Maturity levels

Nonparametric correlations were performed to determine at which maturity level an organisation would be classified to fall under for each system development methodology in use. Table 4.38 shows the correlation coefficients per maturity level. These correlations are significant between two levels:

- Correlation is significant if the Sig. (2-tailed) level is between 0.01 and 0.05 denoted by "**" for a large practical significant relationship a "*" for a medium, practical visible relationship.
- SDLC: As expected, SDLC - being a traditional methodology with formal procedures to be followed - should satisfy all levels of maturity, and this is reflected in the results.
- IE: Information Engineering use resulted in level 3 classifications.
- Agile system development methodologies: Most of the agile methodologies did not result in any level of classification. However, it was discovered that RAD, DSDM and ASD deployment resulted in level 3 and 4 classifications.

  The effect size or practical significance of the correlations between methodologies and maturity levels is represented by the correlation coefficient or Spearman's rho. Spearman's rho indicates the practical significance of the relationship between agile systems development methodologies and maturity levels. A correlation coefficient of 0 indicates that there is no effect; whereas a value of 1 means that there is a perfect effect. The three values below indicate that if the correlation coefficient is (Steyn, 2002):

- ~0.1: Indicates a small effect and thus a non-practical significant relationship exists. For instance, the Scrum methodology at all maturity levels has a relatively small effect thus only a small practical significant relationship exists.
- ~0.3: indicates that there is a medium, practical visible relationship, i.e. Rapid application development and maturity level 3 have a correlation coefficient of 0.296 (almost 0.3), indicating a medium effect relationship.
- ~0.5: indicates that a large practical significant relationship exists. For example, in Table 4.38, SDLC correlation coefficient with maturity level 2 is 0.504. This means that there is a large practical significant relationship between the two.

The practical significant relationships are indicated in Table 4.38 below.

Table 4.38: Correlations between SDMs and Maturity levels

Correlations between SDMs and Maturity levels				
		MATURITY LEVEL		
METHODOLOGIES		Level 2	Level 3	Level 4
SDLC	Correlation Coefficient	.504	.492	.476
	Sig. (2-tailed)	.000**	.000**	.000**
	N	76	75	74
IE	Correlation Coefficient	.166	.289	.194
	Sig. (2-tailed)	.226	.032*	.155
	N	55	55	55
RUP	Correlation Coefficient	.006	.178	.010
	Sig. (2-tailed)	.967	.192	.940
	N	55	55	55
SCRUM	Correlation Coefficient	.130	.189	.173
	Sig. (2-tailed)	.304	.134	.175
	N	64	64	63
RAD	Correlation Coefficient	.194	.296	.452
	Sig. (2-tailed)	.128	.018*	.000**
	N	63	63	62
XP	Correlation Coefficient	.107	.115	.221
	Sig. (2-tailed)	.401	.365	.082
	N	64	64	63
LSD	Correlation Coefficient	.140	.185	.226
	Sig. (2-tailed)	.313	.180	.100
	N	54	54	54
FDD	Correlation Coefficient	.124	.235	.241
	Sig. (2-tailed)	.358	.079	.074
	N	57	57	56
CRYSTAL METHODOLOGIES	Correlation Coefficient	.043	.220	.201
	Sig. (2-tailed)	.757	.109	.145
	N	54	54	54
DSDM	Correlation Coefficient	.119	.276	.297
	Sig. (2-tailed)	.397	.045*	.031*
	N	53	53	53
ASD	Correlation Coefficient	.258	.288	.292
	Sig. (2-tailed)	.060	.034*	.032*
	N	54	54	54

OTHER METHODOLOGIES	Correlation Coefficient	-.114	-.215	-.221
	Sig. (2-tailed)	.463	.162	.155
	N	44	44	43

4.9 Relationship between process/product success factors and maturity levels

In section 4.6.3.1 and 4.6.4.1, we established process success variables and product success variables respectively. The process factors divided into two components can either yield tangible benefits or intangible benefits. Similarly, the product success factors provide short-term and long-term benefits. Table 4.39 illustrates the correlations between the factors and the maturity levels.

A company ought to attain a maturity level 4 in order to enjoy both tangible and intangible benefits. Short-term goals can also only be achieved if the maturity level is level 4. However, long-term benefits can be achieved at any maturity level.

The practical significance of the correlations between process/product success factors and maturity levels is also represented by the correlation coefficient or Spearman's rho (Steyn, 2002). From Table 4.39, we can deduce that:

- ~0.1: Indicates that product short-term goals have a non-practical significant relationship at almost all levels of maturity.
- ~0.3: indicates that there is a medium, practical visible relationship for system supporting factors at maturity level 3.
- ~0.5: indicates that systems support has a large practical significant relationship at maturity level 4 3.

Table 4.39: Correlations between success factors and maturity levels

Relationship of success factors and maturity levels				
		Maturity Levels		
Success Factors(Benefits)		**Level2**	**Level3**	**Level4**
Process triple constraint management (Process_tangible influencing factors)	Correlation Coefficient	.050	.201	.268*
	Sig. (2-tailed)	.679	.096	.027
	N	70	70	68
Process objectives and goals (Process_Intangible influencing factors)	Correlation Coefficient	.202	.160	.257*
	Sig. (2-tailed)	.093	.187	.034
	N	70	70	68
System and goal success (Product_Shortterm goals)	Correlation Coefficient	.147	.197	.266*
	Sig. (2-tailed)	.215	.095	.025
	N	73	73	71
System support (Product_Longterm goals)	Correlation Coefficient	.235*	.321**	.441**
	Sig. (2-tailed)	.045	.006	.000
	N	73	73	71

4.10 CHAPTER SUMMARY

This chapter reported the findings from the data collected via questionnaires distributed to IT professionals from various organisations. The findings of a questionnaire survey, aid to investigate the existence of a relationship (or not) between Agile systems development methodologies (ASDMs) and Software process improvement models (SPIMs).

One hundred questionnaires were gathered by the researcher. The survey presents the results of the analysis of the following aspects: demographic information, critical success factors, the use and benefit of systems development methodologies, the reasons for not using systems development methodologies, and project outcomes. The primary analysis techniques used include: Frequency tables; Crosstabs which aid in

linking the two aspects of the research (ASDMs and SPIMs); and Correlations and factor analysis.

The study provides consistent findings and some interesting correlations and as such we can conclude that the variables in this study provide proof of a pattern.

CHAPTER 5

DISCUSSIONS

5.1 INTRODUCTION

The purpose of this chapter is to discuss the findings obtained from the survey conducted in Chapter 4 by means of various statistical analyses. This chapter will discuss the findings or research objectives which were the basis of this study. The purpose of the study was to establish whether there is a relationship between agile system development methodologies (ASDMs) and software process improvement models (SPIMs). The two were firstly studied independently to gauge their uses in the industry. This chapter discusses the uses of ASDMs and SPIMs in the industry and also seeks to discover the existence of a relationship between the use of system development methodologies and software process improvement models. The chapter further identifies and outlines the current limitations faced by the industry. Last but not least, we propose possible future work that can be done on this topic.

5.2 THE DEMOGRAPHIC INFORMATION

The data analyses show that the respondents were mostly software developers: 52 out of the 100 respondents (52% indicated that they were system developers; 23% were Project Managers/Leaders; and Systems Analysts represented 13% of the respondents. The remaining respondents specified other job category.

Most of the respondents (54%) worked for organizations whose core business area was system development; 43% were from organisations with more than 1000 employees. Forty-two percent of the organisations' information systems departments had 50 or more employees in their IS department. Eighteen percent of the respondents spent 50% of their time to develop new applications, while only 5.2% devoted all their time to developing new applications. The size of projects that respondents had been involved in were predominantly large (47%), and most took less than a year to complete.

The next section will discuss the results obtained from analyses of the data. Each of the successive sections will discuss results for each one of the research objectives established in Chapter 1. As a recap, in order to achieve the primary objective, the research questions set out to assist in reaching the objectives of this study are:

- Objective 1: Study the use and effectiveness of ASDMs in the industry.

- Objective 2: Study the use and effectiveness of SPIMs in the industry.
- Objective 3: Study the relationship between the use of ASDMs and SPIMs

5.3 RESEARCH OBJECTIVE 1: The use and effectiveness of ASDMs

This section focused on the types of, and the extent to which, systems development methodologies are used within the respondents' organisation. The results obtained were summarised with the aim of achieving Objective 1, namely the use and effectiveness of ASDMs in the industry.

5.3.1 To what extent does IS department use standard system development methodologies?

As Table 4.12 indicates, the traditional system development life-cycle is still quite popular, as the results indicate 31.8% 'high extent of use'. RAD and Scrum are next, with a high intensity of use at 14.9% and 13.3% respectively. There is a 32% indication of high use of other methodologies as specified by the respondents, which shows that there are other individual preferred methods. Crystal Methodologies has a high percentage (65.1%) of respondents who did not use it at all. FDD, IE, RAD and Scrum to some extent show an average level of usage.

The results show that SDLC is used to some extent. Other methodologies like RAD and Scrum also show a significant amount of usage; however, the use is not as extensive.

5.3.1.1 Reasons for not using systems development methodologies

In Table 4.18, only 31 respondents disclosed their experience of not using SDMs. Fourteen out of 31 respondents stated that they had not considered using SDMs in their IS department. Ten of the respondents stated that the IS department had considered using SDMs, however for some reason(s) decided against it, while seven respondents disclosed that they previously made use of an SDM, however the SDM had been abandoned.

Table 4.19 lists 13 probable reasons contributing to non-use of SDMs. It emerged that the main contributing reason for organisations not adopting SDMs was that the current systems development practices within the IS department were adequate. This was followed closely by reason number two: the benefits of SDMs being long-term whilst

costs incurred are short-term, and that experienced developers reduced the need for SDMs.

In addition, Table 4.20 presents factor analysis using un-rotated factor loadings to classify reasons for non-use of SDMs into groups of four factors namely: F1, F2, F3 and F4, where F1 represents Uncertainty of benefits and complexity of SDMs, F2: Knowledge deficiency, F3: Direct cost and expertise availability and, lastly F4: Incompatibility and contentment. The reliability of F1 (CA=0.86) and F2 (CA=0.84) is considered good, while F3 (CA=0.56) and F4 (CA=0.53) have low reliability values. Thus the components of the two factors can be evaluated in their individual capacities.

5.4 RESEARCH OBJECTIVE 2: The use and effectiveness of SPIMs

The results from section investigate whether an organisation has been accredited to an industry model of standardisation in the form of Software Process Improvement Models (SPIMs). The results capture the types of (and the levels to which) SPIMs are used within the organisation, as well as descriptions of development procedures and processes for certified organisations.

5.4.1 Software Process Improvement Model (SPIMs) usage

A large contingent of respondents did not know whether the organisation was certified with a process model. Twelve respondents acknowledged having ISO 900-3 in their organisation, whereas only seven reportedly used CMMi (with only one organisation having attained level-5 certification). SPICE was the least used, with only two respondents saying they used it. Thirty-six respondents indicated that no certification with a process model existed in their organisation.

The main motivating factors for certification was to make a better impression in the market, and to produce better products. The presence of a system development methodology was not an influence in the type of process model an organisation would opt for.

5.4.2 SPI procedures and processes

An evaluation of performed processes and procedures within organisations was done.

Table 4.28 lists the procedures and process which are classified into four different classes or levels (level2, level3, and level4). Each process or procedure is an activity that should ideally be performed at a certain level. To compute the mean and standard deviation, the formula in Figure 4.1 was used.

Algorithm in Figure 4.2 classifies the maturity level an organisation falls under, based on the processes and procedures performed. Table 4.27 indicates that 79.2% of the respondents performed activities that are classified at maturity level one, 8.9% at level 2, 3.6% at level 3, and 8.3% did activities that would classify them at maturity level 4.

Crosstabs (Table 4.28) were computed to determine the percentages of these activities performed at each maturity level. A common occurrence observed was that activities were performed haphazardly. For instance, an activity such as having a mechanism for controlling changes to the software design is performed at class level 3 – yet the results show that no organisations at level 3 performed it. However, the results also indicate that 73.3% and 93.8% (representing level 1 and level 2 respectively) performed the activity - which is above their classification. Thus, it is concluded that the practices in the industry do not follow the stipulated standards.

5.4.3 Factors contributing to non-SPIM certification

In Table 4.29, 20 respondents indicated that the lack of benefits or advantages were the main factor why their organisation was not certified with a process model. Affordability was cited nine times as a factor. The lack of benefit, skill and tools did not seem to be a major contributing factor.

5.5 RESEARCH OBJECTIVE 3: Project Outcomes

A total of 98 respondents described the outcome of the last project. An overwhelming 86 respondents said that the project was completed and implemented and still in use. Six projects were cancelled before completion, and only one completed and implemented project had ceased.

The longest period that a complete and implemented project had been in use is 120 months. The rest only had some months in use, of which nearly 15% had only been

rolled out for six months.

5.5.1 Process factors affecting project success rate

These factors aimed to illustrate that project management success is measured by being able to manage the knowledge areas of project management, especially those that form part of the triple constraint. That is, the ability to manage the cost, scope and schedule of the project whilst maintaining high quality standards of the end product. Table 4.32 presents these process factors. A project that achieves its goal will have a high success rating.

Furthermore, a factor analysis was performed where various process success variables are grouped into two un-rotated factor loadings to determine the nature of the construct influencing observed process success variables. In Table 4.33, the outcomes of the project are tabulated using two factor loadings (F1 and F2). F1 comprises of variables that reflect the scope, time and cost management processes, so it is named “Process triple constraint management”. F2 comprises variables that illustrate the goal achievement process, so it is referred to as “process objectives and goals”. The reliability of the factors was measured by means of Cronbach’s alpha (CA). F1 and F2’s Cronbach’s alphas measured 0.77 and 0.82 respectively. Given that both CAs are above the industry recommend value of 0.70, it reflects that the reliabilities are good, therefore proof that the validity of the variables is good. F1 and F2’s Cronbach’s alphas measured 0.77 and 0.82 respectively. Given that both CAs are above the industry recommended value of 0.70, it reflects that the reliabilities are good, thus proof that the validity of the variables is good.

5.5.2 Project success factors in terms of system scope

In Table 4.34, factors affecting the scope of the actual system that was developed in terms of functionality, maintainability, reliability, and efficiency are tabulated. The results show that the developed system meets user’s needs, with 84.5% of the respondents in agreement. Generally, the respondents agreed with all the factors.

As was the case with process factors, a factor analysis and reliability test was performed to determine the nature of the construct influencing observed system scope success variables. Table 4.35 measures the outcome of the project using two factor

loadings (F1 and F2). F1 comprises of variables that are aligned to the effectiveness of the system and thus an indication of whether the intended goal has been achieved. These variables can be grouped under one factor category as “System and goal success”. Whilst F2 comprises variables that play a supportive role to the system once it has been rolled out, such as system documentation, portability and maintainability, thus we categorize them as “System support”. F1 and F2’s Cronbach’s alphas measured 0.92 and 0.71 respectively. Both CA’s indicate that the reliabilities are good, also an indication of a good validity of the variables.

5.6 Combination of SDMs and SPIMs use in Organisation

Crosstabs were performed per SDM against each process model (Table 4.37) to gauge how much a system development methodology is used when a certain software process improvement model is in place... The majority of the SDMs combined well with ISO-900-3. However, in most instances, the absence of a process model (non-certification) resulted in most respondents indicating that they did use that particular methodology. The highest number of responses was when the respondents were not aware of the presence of process improvement model within the organisation.

The effect size or practical significance of crosstabs between the extents of SDM use per SPIM is represented by phi coefficient or Cramer’s V (presented in Table 4.37).

- If phi is~0.1 then there is a practical non-significant association between the variables.
- If phi is ~0.3, then the relationship is a practical visible significant association and,
- If phi is~0.5, then the association is practically significant or a large effect.

From the results, it can be concluded that most respondents were either non-certified or simply not aware of the combinational use of SDMs and SPIMs.

5.6.1 The Correlations between SDMs and Maturity levels

Nonparametric correlations were performed to determine at which maturity level a company would be classified for each system development methodology in use.

Table 4.38 shows the correlation coefficients per maturity level. SDLC, being a

traditional methodology with formal procedures to be followed, should satisfy all levels of maturity as is the case in the results. Most the agile methodologies did not result in any level of classification. However, it was discovered that RAD, IE, DSDM and ASD deployment resulted to level 3 and 4 classification.

5.6.2 Relationship between process/product success factors and maturity levels

The process factors provide tangible and intangible benefits, while product success factors provide short-term and long-term benefits. In Table 4.39, the correlations between the factors and the maturity levels show that an organisation ought to attain a maturity level 4 in order to enjoy both tangible and intangible benefits. Short-term goals can also only be achieved if the maturity level is level 4. However, long-term benefits can be achieved at any maturity level.

5.7 ACADEMIC CONTRIBUTION OF THE RESEARCH

The study uncovered that agile systems development methodologies are compatible in theory with software process improvement models. This interrelationship has it potential benefits, such as an accelerated development process that delivers better-quality software. However, to make the combinations possible, alterations are made to both methodology and model. This could possibly be the birth of a new range of methodologies. An organisation wanting to improve the software-development process and be mature, can take on board the results of this study to aid in making that decision. From the literature, we highlighted some benefits an organisation can enjoy by combining agile methods and the software process model.

5.8 LIMITATIONS AND FUTURE WORK

More insight would have been gained with a larger pool of data sources. The use of questionnaires, as effective as they can be, could have been adequately supported in conjunction with other research strategies, such case studies. Due to challenges encountered with a slow response rate, the 100 returned questionnaires (that is, responded to) were adequate - but not as many as the researcher would have liked.

In terms of future work, more research needs to be done with regards to other process

models other than CMMi, to also have an in-depth view on their possible co-existence with agile methodologies. Organisations are getting more agile and enhancing customer satisfaction; however, more needs to be done with regards to the co-existence and benefits gained from certification, to improve software development.

5.9 SUMMARY

This chapter provided a summary of the findings conducted from the questionnaire data. In contrast to theoretical findings, interpretation of the results indicates that although organisations are becoming more agile in their system development process, it is rare in practice that an organisation would implement a combination of ASDMs and SIPIMs. However, the combination may yield numerous benefits for the organisation.

BIBLIOGRAPHY

Abrahamson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. Agile software development methods. Review and Analysis. VTT Publications 478. 107p.

Alegrıa, J.A.H. and Bastarrica, M.C. 2006. Implementing CMMi using a combination of agile methods. CLEI electronic Journal, 9(1):1-15.

Alexandrou, M. 2011. Lean Development (LD) Methodology. Mariosalexandrou.com. http://www.mariosalexandrou.com/methodologies/lean-development.asp. Date of access: 09th June 2015.

al-Tarawneh, M.Y., Abdullah, M.S. & Ali, A.B.M., 2011. A proposed methodology for establishing software process development improvement for small software development firms. Procedia Computer Science, 3:893-897.

Ambler, S.W. 2005. The Agile Unified Process (AUP). Ambysoft. http://www.ambysoft.com/unifiedprocess/agileUP.html. Date of access: 12th July 2014.

Ambler, S.W. 2009. Agile Software Development by the Numbers: What's Really Going On Out There. IBM Rational Software Conference. ftp://public.dhe.ibm.com/software/in/rsc2009/2_Agile_by_the_Numbers-Scott_Ambler.pdf. Date of access 7th December 2015.

Avison, D. & Fitzgerald, G. 2006. Information Systems Development: Methodologies, Techniques and Tools. 4th Edition. United Kingdom: McGraw-Hill Education.

Awad, M. A. 2005. A Comparison between Agile and Traditional Software Development Methodologies.

Babbie, E. 2004. The Practice of Social Research (10th ed). Wadsworth: Chapman University. 640 p.

Bamford, R., C. & Deibler II, W., J. 1993. Comparing, contrasting IS0 9001 and the SEI capability maturity model. IEEE Computer, 26(10): 68-70, Oct.

Beck, K. 1999. Extreme Programming Explained: Embrace Change. Reading, MA; Addison-Wesley.

Bella, F.,Hormann, K. & Vanamali,B. 2008. From CMMi to SPICE – Experiences on How to Survive a SPICE Assessment Having Already Implemented CMMi, 5089:133-142.

Beck, K., *et al.* 2001. Manifesto for Agile Software Development. http://www.agilemanifesto.org/. Date of access. 14th March 2014.

Boehm, B. & Turner, R. 2005. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. IEEE Software, 22(5): 30-39, Sep/Oct.

Boehm, B. & Turner, R. 2003. Balancing Agility and Discipline - A guide for the perplexed. Addison-Wesley.304p.

Bryman, A. 2004. Qualitative research on leadership: A critical but appreciative review. The Leadership Quarterly. 15: 721-891.

Bryman, A. & Cramer, D. 2002. Quantitative data analysis with SPSS release 8 for Windows: a guide for social scientists. Routledge.304p.

Chan, F.K.Y. & Thong, J.Y.L. 2009. Acceptance of agile methodologies: A critical review and conceptual framework.

CMMi Architecture Team. 2007. Introduction to the Architecture of the CMMi Framework, Technical Note, Report CMU/SEI-2007-TN-009, Software Engineering Institute, Pittsburgh.

CMMi Product Team. 2002. Capability Maturity Model Integration (CMMi) Version 1.1, Continuous Representation, Report CM U/SEI-2002-TR-011, Software Engineering Institute, Pittsburgh.

Cohen, D., Lindvall, M. & Costa, P. 2004. An introduction to agile methods. Advances in computers, 62:1-66.

Cohen, J. 1988. Statistical power analysis for the behavioural sciences. Second Edition Hillsdale, NJ: Erlbaum. 567 p.

Coallier, F. 1994. How ISO 9001 fits into the software world. *IEEE Software*.11 (1):98-100.

Conboy, K. 2009. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development, Information Systems Research, 20 (4): 329-354.

Creswell, J. W. 1994. Research Design: Qualitative and Quantitative Approaches. Thousand Oaks. CA: Sage. 228 p.

Creswell, J. 2003. Research design: Qualitative, quantitative and mixed methods approaches. 2nd ed. Thousand Oaks, CA: Sage. 246 p.

Dalalah, A., 2014. Extreme Programming: Strengths and Weaknesses. Computer Technology and Application, 5(1).

Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N.B., 2012. A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6): 1213-1221.

Dybå, T. & Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. Information and software technology, 50(9):833-859.

Ehsan, N., Perwaiz, A., Arif, J., Mirza, E. & Ishaque, A. 2010. "CMMi/SPICE based process improvement," in *Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference on:* 859–862.

Ellis, S.M. & Steyn, H.S. 2003. Practical significance (effect sizes) versus or in combination with statistical significance (p-values), Management Dynamics, 12(4): 51-53.

Emam, K.E, Melo, W & Drouin,J.N. 1998. SPICE: The theory and practice of software process improvement and capability determination. IEEE Computer Society Press.496P.

Emam, K.E & Jung, H.,-W. 2001. An empirical evaluation of the ISO/IEC 15504 assessment model. Journal of Systems and Software, 59(1): 23-41, Oct.

Eysenck, M. 2004. Research Methods: Data Analysis. Psychology Press Ltd.

Fayad, M.E, & Laitnen, M. 1997. "Process assessment considered wasteful," Communications of the ACM, 40(11): 125-128.

Fojtik, R. 2011. Extreme Programming in development of specific software. Procardia Computer Science. 3: 1464-1468.

Fritzsche, M. & Keil, P. 2007. Agile Methods and CMMi: Compatibility or Conflict?, e-Informatica Software Engineering Journal, 1(1): 9-26.

Gable, G.G. 1994. "Integrating Case Study and Survey Research Methods: An Example in Information Systems". European Journal of Information Systems, 3(2):112-126.

Garg, A. 2009. Agile Software Development. DRDO Science Spectrum. 55-59.

Gefen, D., Zviran, M. & Elman, N. 2006. What Can be Learned from CMMi Failures? Communications of the Association for Information Systems,17:801-817.

Glazer, H., Dalton, J., Anderson, D., Konrad, M. & Shrum,S. 2008. CMMi® or Agile: Why Not Embrace Both! Software Engineering Process Management.

Halvorsen, C. P. & Conradi, R. 2001. A Taxonomy to Compare SPI Frameworks. Lecture notes in computer science, 217-235.

Herbsleb, J.D, Zubrow, D., Goldenson, D., Hayes, W. & Paulk, M.C. 1997. Software Quality and the Capability Maturity Model. Commun. ACM, 40(6):30–40.

Highsmith, J.A. 2000. Adaptive Software Development: A Collaborative Approach to Managing Complex systems. New York: Dorset House.

Highsmith, J.A. 2002. Agile Software Development Ecosystems. Addison-Wesley. 404p.

Highsmith, J.A. 2001. History: The Agile Manifesto. http://www.agilemanifesto.org/history.html. Accessed March 28, 2010.

Höggerl, M. and Sehorz, B. 2006. “An Introduction to CMMi and its Assessment Procedure”, Seminar for Computer Science, Department of Computer Science University of Salzburg.

Hopkins, W.G. 2002. Dimensions of Research. *Sportscience*. http://www.ww.sportsci.org/jour/0201/wghdim.htm. Date of access: 18th July 2011.

Ho-Won, J. & Hunter, R. 2001.The relationship between ISO/IEC 15504 process capability levels, ISO 9001 certification and organization size: An empirical study. Journal of Systems and Software, 59(1):43-55.

Huang, S. & Han, W. 2005. Selection priority of process areas based on CMMi continuous representation. Information & Management, 43(2006): 297–307.

Huisman, H.M. & Iivari, J. 2006. Deployment of systems development methods: Perceptual congruence between IS managers and system developers. *Information & Management,* 43: 29-49.

Humphrey, W. 1991. Recent findings in software process maturity. Software Development Environments and CASE Technology, 509(1991): 258-270.

ICMIT, Islamabad, Pakistan: 859-862.

Humphrey, W.S., 1988. Characterizing the software process: a maturity framework. IEEE software, 5(2): 73-79.

Iivari, J. & Iivari, N. 2010. The relationship between organizational culture and the deployment of agile methods. Information and Software Technology.

Jeffries, R, Anderson, A. & Hendrickson, C. 2001. Extreme programming installed. Harlow: Pearson Education.

Kehoe, R. & Alka, J. 1996. ISO 9000-3: A tool for software product and process

Kehoe, R., & Jarvis, A. 1996. *ISO 9000-3: A tool for Software Product and Process Improvement. Edition en anglais*. Springer Science & Business Media.

Ketter, W., Banjanin, M., Guikers, R. & Kayser, A. 2009. Introducing an Agile Method for Enterprise Mash-Up Component Development. IEEE Conference on Commerce and Enterprise Computing.

Kim, S. 2003. Research paradigms in organizational learning and performance: Competing modes of inquiry. Information technology, learning, and performance journal, 21(1):9-18, Spring.

Koutsoumpos, V. & Marinelarena, I. 2013. Agile Methodologies and Software Process Improvement Maturity Models, Current State of Practice in Small and Medium Enterprises. Karlskrona, Sweden: Blekinge Institute of Technology. (Dissertation - M.Sc.). 230p

Leedy, P. & Ormrod, J. 2001. Practical research: Planning and design. 7th ed. NJ: Sage. 317 p.

Leithiser, R. & Hamilton, D. 2008. Agile versus CMMi - process template selection and integration with micro-soft team foundation server. Proceedings of the 46th Annual Southeast Regional Conference on XX. New York: 186-190.

Lina, Z. & Dan, S. 2012. “Research on Combining Scrum with CMMi in Small and Medium Organizations”. International Conference on Computer Science and Electronics Engineering (ICCSEE), 1:554–557.

Manders, B., de Vries, H.J. & Blind, K. 2016. ISO 9001 and product innovation: A literature review and research framework. Technovation, 48-49:41-55.

Matveev, A.V. 2002. The advantages of employing quantitative and qualitative methods in intercultural research: practical implications from the study of the perceptions of intercultural communication competence by American and Russian managers. Institute of Management, Business and Law Publishing: 59-67.

Marcal, A.S.C., Soares, F.S.F. & Belchior, A.D. 2007. Mapping CMMi project management process areas to SCRUM practices. In Software Engineering Workshop, 2007. SEW 2007. 31st IEEE:13-22.

Matsunaga, M. 2015. How to Factor-Analyze Your Data Right: Do's, Don'ts, and How-To's. International Journal of Psychological Research, 3(1): 97-110.

Mazengera, B. 2009. The Use of Systems Development Methodologies in the Telecommunication Industry. Potchefstroom: NWU. (Dissertation - M.Sc.). 230p.

McAdamand, R & Fulton, F. 2002. "The impact of the ISO 9000:2000 quality standards in small software firms", Managing Service Quality: An International Journal, Vol. 12(5): 336-345.

Mellon, C. http://www.sei.cmu.edu/cmmi/: Accessed 28 November 2014.

Meredith, J. 1998. Building operations management theory through case and field research. Journal of Operations Management, 16(4): 441-454, Jul.

Milne, J. 1999. Questionnaires: Some advantages and disadvantages. Centre for CBL in Land Use and Environmental Sciences, Aberdeen University.

Myers, M.D. 1997. Qualitative Research in Information Systems. *MIS Quarterly*. 21(2): 241-242. http://www.qual.auckland.ac.nz/. Date of access: 30th May 2011.

Nahler, G. 2009. ISO 9000-3. *Dictionary of Pharmaceutical Medicine*. Springer Vienna. 100p.

Nguyen, N.T. 2010. How software process improvement standards and agile methods co-exist in software organisations?

Oates, B.J. 2006. Researching Information Systems and Computing. London: Sage. 341 p.

Oates, B.J. 2008. Researching information systems and computing. London: Sage.

Paulk, M. C. 1995. How ISO 9001 compares with the CMM. *Software IEEE*.12:74-83.

Paulk, M.C. 2001. Extreme programming from a CMM perspective. IEEE software,18(6):19-26.

Palmer, S.R & Felsing, J.M. 2002. A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall. 304p.

Patton, M.Q. 2002. Qualitative Research and Evaluation Methods. London: Sage. 598 p.

Pfahl, D. 2008. INF5180: Software Product- and Process Improvement in Systems Development. Part 09: Process Improvement Frameworks.

Pinsonneault, A. & Kraemer, K.L. 1993. Survey Research Methodology in Management Information Systems: An Assessment. Journal of Management Information Systems, 10(2):75-105, 1 Sep.

Poppendieck, M. 2007. Lean software development. In Companion to the proceedings of the 29th International Conference on Software Engineering. IEEE Computer Society: 165-166.

Poppendieck, M. & Poppendieck, T. 2006. Implementing Lean Software Development: From Concept to Cash. Addison Wesley. Pg. 23 – 40.

Poppendieck, M. and Cusumano, M.A. 2012. Lean software development: A tutorial. IEEE software, 29(5):26-32.

Ross, K.N. 2005. Quantitative research methods in educational planning. Paris: UNESCO International Institute for Educational Planning. Available online: http://www.unesco.org/iiep.

Ruiz, J.C., Osorio, Z.B., Mejia, J., Muñoz, M., Ch, A.M. and Olivares, B.A. 2011. Definition of a hybrid measurement process for the models ISO/IEC 15504-ISO/IEC 12207: 2008 and CMMi Dev 1.3 in SMEs. In Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE: 421-426.

Scrum: “What is Scrum? ,” https://www.scrum.org/Resources/What-is-Scrum Accessed March 19, 2014.

Saddle River, NJ, Prentice-Hall.

Seaman, C.B. 1999. Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*. 25(4): 557-572

Simon, JM. 1996. SPICE: Overview for software process improvement. Journal of Systems Architecture, 42: 633-641.

Singels, J., Ruel, G. & van de Water, H. 2001. ISO 9000 series Certification and performance. International Journal of Quality & Reliability Management, 18(1): 62-75.

Soy, S.K. 1997. The Case Study as a Research Method. http://www.ischool.utexas.edu/~ssoy/usesusers/l391d1b.htm. Date of access: 30th May 2015.

Stahl, B. C. 2005. A critical view of the ethical nature of Interpretive research: Paul Ricoeur and the other. European Conference on Information Systems, Paper 29.

Stelzer, D, Mellis, W. & Herzwurm, G. 1996. Software Process Improvement via IS0 9000? Results of two surveys mong European software houses. Proceedings of the 29th Annual Hawaii International Conference on System Sciences.

Steyn, H.S. (jr.). 2000. Practical significance of the difference in means, SA Journal of Industrial Psychology, 26(3), 1-3.

Steyn, H.S. (jr.). 2002. Practically significant relationships between two variables, SA Journal of Industrial Psychology, 28(3), 10-15.

Steyn, H.S. (jr.) 2009. Manual: Effect size indices and practical significance. http://www.nwu.ac.za/content/statcs-manual. North-West University (Potchefstroom Campus), Potchefstroom. Date of access: 12th July 2015.

Stapleton, J. 2003. DSDM Consortium, DSDM: Business Focused Development. 2nd Edition. Addison-Wesley.

Takeuchi, H. & Nonaka, I. 1986. The New Product development Game. Harvard Business Review Jan. /Feb.: 137.

Taylor, P.C. 2006. Major Research Paradigms for Science and Mathematics Educators. Curtin University of Technology. Australia

Therese, C.V., & Alagarsamy, K. 2011. Extreme Programming versus CMMi - Conflicts and Compatibilities. International Journal of Computer Science Engineering & Technolo, 1(5): p203

Walsham, G. 1995. The Emergence of Interpretivism in IS Research. Journal of Information Systems Research, 6(4):376-394, Dec.

Walsham, G. 1995. Interpretive case studies in IS research: nature and method. European Journal of Information Systems, 4(2):74-81.

Walsham, G. 2006. Doing interpretive research. European Journal of Information Systems, 15(3): 320–330.

Wang, Y. & King, G. 2000. Software engineering processes: principles and applications. CRC press.

Williams, C. 2007. Research Methods. Journal of Business & Economic Research, 5(3):65-72, March.

Windholtz, M. 2005. Lean Software Development. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.2561&rep=rep1&type=pdf Accessed March 7, 2010.

Wynekoop, J.L. & Russo, N.L. 1993. Systems development methodologies: unanswered questions and the research-practice gap. In: DeGross, J.I. (Eds.). *Proceedings of the Fourteenth International Conference of Information Systems.* Orlando, Florida: 181-190.

Weaver, P. 2004. Success in your project: a guide to student system development projects. Upper Saddle Hill, NJ: Prentice Hall.

Yin, R.K. 1993. Application of Case Study Research. CA: Sage Publishing: 33-35.

Yin, R.K. 1994. Case Study Research: Design & Methods (2nd Ed). Thousand Oaks, CA: Sage Publications.

ANNEXURES A: RESEARCH QUESTIONNAIRE

Research Questionnaire: The relationship between agile systems development methodologies and software process improvement models

The purpose of this study is to investigate the relationship between agile systems development methodologies (ASDMs) and software process improvement models (SPIMs). The study also focuses on the extent of use of SPIMs and ASDMs in the industry with respect to effectiveness of horizontal and vertical uses within an organization.

Your participation in this study is valuable and be assured that your responses will be kept confidential and the completion of this questionnaire is voluntary. The results of this study will be provided on request.

SECTION A: Organisation Background Information

1. Please indicate your job category.	
1.1. Project Manager/Leader	1
1.2. Developer	2
1.3. Systems Analyst	3
1.4. Other, please specify	4

2. Please indicate the core business area of your organization.	
2.1. Administrative services	1
2.2. Manufacturing	2
2.3. Education	3
2.4. System Development/ Software consulting	4

2.5. Community services	5
2.6. Financial, Banking	6
2.7. Government	7
2.8. Other, please specify	8

3. Please indicate the total number of employees in your Organization (at all locations).	
3.1. 1 – 50 employees	1
3.2. 51 – 200 employees	2
3.3. 201 – 500 employees	3
3.4. 501 – 1000 employees	4
3.5. More than 1000 employees	5

4. Please indicate the total number of employees in your organization's information system (IS) department (at all locations)?	
4.1. 1 – 5 employees	1
4.2. 6 – 20 employees	2
4.3. 20 – 50 employees	3
4.4. More than 50 employees	4

5. What percentage (%) of your IS department's effort is devoted to development of new applications?	%
6. What percentage (%) of your IS department's effort is devoted to systems maintenance and support	%

7. What percentage (%) of your IS department's effort is devoted to package customization	%

8. Please indicate the size of the last information system development project you were involved with.	
8.1. Small	1
8.2. Medium	2
8.3. Large	3
8.4. Very Large	4
8.5. None partaken	5

9. Please indicate the duration of the last project you were involved with.	
9.1. Less than a year	1
9.2. 1 – 2 years	2
9.3. 3 – 5 years	3
9.4. 6 years or more	4

SECTION B: Systems Development Methodologies used

1. To what extent does your IS department use the following standard (commercial) system development methodologies at present? You may mark more than one item.(1=Nominally 5=Intensively)	**1=Nominally**				**5=Intensively**
1.1. Systems Development Life Cycle (SDLC). 1.2. Information Engineering (IE)	1	2	3	4	5

1.3. Rational Unified Process (RUP)	1	2	3	4	5
1.4. Scrum	1	2	3	4	5
1.5. Rapid Application Development (RAD)	1	2	3	4	5
1.6. Extreme Programming (XP)	1	2	3	4	5
1.7. Lean Software Development (LSD)	1	2	3	4	5
1.8. Feature-Driven Development (FDD)	1	2	3	4	5
1.9. Crystal Methodologies	1	2	3	4	5
1.10.Dynamic Systems Development Methodologies (DSDM)	1	2	3	4	5
1.11. Adaptive Software Development (ASD)	1	2	3	4	5
1.12. Other, please specify	1	2	3	4	5

If IS department has NO systems development methodology in place, please skip to Question 7

2. How long has your systems development methodology been in use in your IS department?	
2.1 Less than a year	1
2.2. 1 – 2 years	2
2.3. 3 – 5 years	3
2.4. 6 – 10 years	4
2.5. 11 years or More	5
2.6. I don't know	6

3. What motivated the choice of the systems development methodology used in your IS department?	
3.1. Increased productivity and quality	1
3.2. Software process improvement certification	2
3.3. Standardizing the development process	3

3.4. I don't know	4
4. What is the proportion of projects that are developed in your IS department by applying systems development methodology knowledge?	
4.1. None 4.2. 1 – 25% 4.3. 26 – 50 % 4.4. 51 – 75 % 4.5. 76 % or More	1 2 3 4 5

5. What is the proportion of people in your IS department that apply systems development methodology knowledge regularly?	
5.1. None 5.2. 1 – 25% 5.3. 26 – 50 % 5.4. 51 – 75 % 5.5. 76 % or More	1 2 3 4 5

6. Which of the following best describes how your IS department makes use of its systems development methodologies?	
6.1. A Standard which is followed rigorously for all projects 6.2. A general guideline for all projects. 6.3. Adapted on a project-to-project basis	1 2 3

If your IS department does not use any systems development methodologies (commercial or in-house developed) as part of system development methodology, please answer the following questions.

7. Which of the following statements describe the situation in your IS department?	
7.1. Your IS department had never considered using systems development methodologies	1
7.2. Your IS department had considered using systems development methodologies, but decided against it.	2
7.3. Your IS department did use systems development methodologies in the past, but abandoned it.	3

8. To what extent do you agree/disagree with the following statements about the last project you were involved with?	Totally Disagree	Disagree	Neutral	Agree	Totally Agree
8.1. The profile of development projects in our IS department doesn't require the use of systems development methodologies.	1	2	3	4	5
8.2. Systems development methodologies are too complex or hard to use.	1	2	3	4	5
8.3. The current systems development practice in our IS department is adequate.	1	2	3	4	5
8.4. The experience of the developers in our IS department reduces the need for systems development methodologies.	1	2	3	4	5
8.5. The benefits of systems development methodology use are long-term, whereas costs are incurred short term.	1	2	3	4	5
8.6. There is a lack of experienced staff in our IS department who can					

effectively use systems development methodologies.	1	2	3	4	5
8.7. New systems developed with systems development methodologies are not compatible with legacy systems.	1	2	3	4	5
8.8. Our IS department lacks a suitable environment to support systems development methodologies.	1	2	3	4	5
8.9. In our IS department there is a lack of management support for the use of systems development methodologies.	1	2	3	4	5
8.10. The learning curve for systems development methodologies is very long.	1	2	3	4	5
8.11. The financial investment in systems development methodologies is too large.	1	2	3	4	5
8.12. In our IS department there is a lot of uncertainty over the benefits of adopting systems development methodologies.	1	2	3	4	5
8.13. In our IS department there is no clear objectives for adopting systems development methodologies.	1	2	3	4	5

SECTION C: Software Process Improvement Model Certification

1. Which of the following Software Process Improvement Models (SPIM) is your organisation certified with? (If any)	
1.1. CMMi	1
1.2. ISO 9000-3	2

1.3. SPICE	3
1.4. No Certification	4
1.5. I don't know	5
1.6. Other, please specify	6

2. If your organisation is CMMi certified, Please specify the level of certification.(i.e. mark level with an **X)**	1	2	3	4	5

If you selected options 1.1, 1.2 or 1.3 from question 1 of Section C above, please complete the following question:

3. What motivated your organisation to become certified?	
3.1 Better market impression	1
3.2 Better quality products	2
3.3 Faster development times	3
3.4 The required skills/tools are already available	4
3.5 Other (please specify):	5

4. Did the IS department's system development methodology have an influence on the type of certification in place?	YES	NO
Please motivate your answer:		

5. Please describe the development procedures and processes in your IS department.		
5.1. Is a formal procedure used in the management review of each software development project prior to making contractual commitments?	YES	NO

5.2. Is a formal procedure used to make estimates of software size?	YES	NO
5.3. Is a formal procedure used to produce software development schedules?	YES	NO
5.4. Is a formal procedure used to make estimates of software development cost?	YES	NO
5.5. Do software development first-line managers sign off on their schedules and cost estimates?	YES	NO
5.6. Does senior management have a mechanism for the regular review of the status of software development projects	YES	NO
5.7. Is there a software configuration control function for each project that involves software development	YES	NO
5.8. Are profiles of software size maintained for each configuration items, over time?	YES	NO
5.9. Is a mechanism used for controlling changes to the software requirements? (Who can make changes and under which circumstances?)	YES	NO
5.10. Is a mechanism used for controlling changes to the software design? (Who can make changes and under which circumstances?)	YES	NO
5.11. Is a mechanism used for controlling changes to the code? (Who can make changes and under which circumstances?)	YES	NO
5.12. Is there a software engineering process group function?	YES	NO
5.13. Does your IS department use a standardized software development process?	YES	NO
5.14. Does your IS department use a standardized and documented software development process on each project?	YES	NO
5.15. Is a mechanism used for ensuring compliance with the software engineering standards?	YES	NO

5.16. Is there a required software engineering program for software developers?	YES	NO
5.17. Is a formal training program required for design and code review leaders?	YES	NO
5.18. Does the Software Quality Assurance (SQA) function have a management reporting channel separate from the software development project management?	YES	NO
5.19. Are internal software design reviews conducted?	YES	NO
5.20. Are the action items resulting from design reviews tracked to closure?	YES	NO
5.21. Are statistics on software design errors gathered?	YES	NO
5.22. Are software code reviews conducted?	YES	NO
5.23. Are the action items resulting from code reviews tracked to closure?	YES	NO
5.24. Are statistics on software code and test errors gathered?	YES	NO
5.25. Are design errors projected and compared to actual?	YES	NO
5.26. Are the review data gathered during design reviews analysed?	YES	NO
5.27. Are code and test errors projected and compared to actual?	YES	NO
5.28. Are the error data from code reviews and tests analysed, to determine the likely distribution and characteristics of errors remaining in the product?	YES	NO
5.29. Are design and code review coverage measured and recorded?	YES	NO
5.30. Is review efficiency analysed for each project?	YES	NO
5.31. Are code review standards applied?	YES	NO
5.32. Is test coverage measured and recorded for each phase of functional testing?	YES	NO
5.33. Is there a mechanism for assuring the adequacy of regression testing?	YES	NO

5.34. Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?	YES	NO
5.35. Has a managed and controlled process database been established for process metrics data across all projects?	YES	NO
5.36. Is a mechanism used for periodically assessing the software engineering process, and implementing indicated improvements?	YES	NO
5.37. Are analyses of errors conducted to determine their process related causes?	YES	NO
5.38. Is a mechanism used for managing and supporting the introduction of new technologies?	YES	NO

If you are NOT CERTIFIED (i.e. did not select options 1.1, 1.2 or 1.3 from question 1 of Section C above), please complete the following question:

6. Why is your organisation not certified?	
6.1 Does not meet required criteria	1
6.2 There are no benefits or advantages	2
6.3 Lack of funds	3
6.4 The required skills/tools are not available	4
6.5 Other (please specify):	5

SECTION D: Project Outcome

1. Which of the following statements describe the outcome of the last system development project you were involved with?	
1.1. The project was cancelled/terminated before time.	1
1.2. The project was completed but not implemented.	2
1.3. The project was completed and implemented, but not in use anymore.	3
1.4. The project was completed and implemented, and is still in use.	4

2. If you have selected **1.4** in the previous question, for how many months has it been in use?	

If your last project were cancelled before completed, you have completed the questionnaire.

THANK YOU FOR YOUR TIME

If your last project was not cancelled before completion (i.e. you selected 1.2, 1.3 & 1.4 in question 1), please answer the following questions.

3. To what extent do you agree/disagree with the following statements about the last project you were involved with?	Totally Disagree	Disagree	Neutral	Agree	Totally Agree
3.1. The project was completed on schedule.	1	2	3	4	5
3.2. The project was completed within the budget.	1	2	3	4	5
3.3. The developed system satisfied all the stated requirements.	1	2	3	4	5
3.4. Speed of developing new applications was high.	1	2	3	4	5
3.5. The productivity involved in the projects was high.	1	2	3	4	5
3.6. The cost of project is low when compared to the size and complexity of the system developed.	1	2	3	4	5
3.7. The project achieved its goal.	1	2	3	4	5
3.8. Overall, the project represents excellent work.	1	2	3	4	5
3.9. Overall, the project was success.	1	2	3	4	5

4. To what extent do you agree/disagree with the following statements about the last project you were involved with?	Totally Disagree	Disagree	Neutral	Agree	Totally Agree
4.1. The functionality of the developed system is high.	1	2	3	4	5
4.2. The reliability of the developed system is high.	1	2	3	4	5
4.3. The maintainability of the developed system is high.	1	2	3	4	5
4.4. The portability of the developed system is high.	1	2	3	4	5
4.5. The efficiency of the developed system is high.	1	2	3	4	5
4.6. The usability of the developed system is high.	1	2	3	4	5
4.7. The developed system meets user's needs.	1	2	3	4	5
4.8. The documentation of the developed system is good.	1	2	3	4	5
4.9. Overall, the quality of the developed system is high.	1	2	3	4	5
4.10. Overall, the users are satisfied with the developed system.	1	2	3	4	5
4.11. Overall, the developed system is a success.	1	2	3	4	5

Any comments**:**

THANK YOU FOR YOUR TIME IN COMPLETING THIS QUESTIONNAIRE!!!