

THE USE AND EFFECTIVENESS OF SYSTEMS DEVELOPMENT METHODOLOGIES IN SOUTH AFRICA

Veemal Kalanjee, B.Sc., Hons B.Sc.

Dissertation submitted in partial fulfilment of the requirements for the degree *Master of Science* at
the Potchefstroom Campus of the North-West University

Supervisor: Dr. H.M. Huisman

January
2006

ACKNOWLEDGEMENTS

I would like to thank Dr. Magda Huisman for the guidance and support that she gave during the duration of this study. The patience and understanding you showed hugely assisted in the completion of this dissertation.

To my parents and sister, thank you for your love and support not only during my study, but through my entire life leading up to this point. I love you guys dearly.

Priya, you were the person that encouraged me to “vasbyt” when I felt like I couldn’t anymore. Thank you for being there when I needed you the most.

A final thank you goes to my colleagues at KID for their understanding. Chris, Derek and Adam, I am finally done with my dissertation and can start doing some work now!

Thank You.

ABSTRACT

This study investigates the use and effectiveness of systems development methodologies in practice. Systems development methodologies are defined as the totality of systems development approaches, process models, methods and techniques used in an organisation. The term “deployment” is used to encompass the use of systems development methodologies, the perceived support provided by them, and their impact on both the developed system and the development process.

The general purpose of this study was to investigate whether there has been a change in the use of systems development methodologies in South Africa since the study performed by Huisman (Huisman, 1999) and also whether the factors that influence the use and effectiveness of systems development methodologies have changed or remained the same as put forth in 1999.

Research was conducted to determine the current situation of systems development methodologies deployment in South Africa. The results showed that development of software is performed on multiple operating systems, using multiple development platforms and multiple programming languages. The requirements and testing phases make use of techniques the most intensively. The overall intensity of use of techniques and methods was found to be very low.

The relationship between the perceived maturity of an organisation and the deployment of systems development methodologies was also investigated. Organisations do not follow the maturity levels of the CMM as prescribed and skip key practices of each level. The CMM Level 3 key practices were found to have the greatest impact on the deployment of methodologies.

Organisational culture and its influence on the deployment of systems development methodologies was another focal point of the analysis. In general, organisations had weak affiliations to the four culture groups as defined by the Competing Values Framework. The culture group with the most impact on the deployment of methodologies was the developmental culture which emphasises decentralisation, differentiation and flexibility.

This is a significant finding since Huisman (1999) found that hierarchical culture most influences deployment of methodologies. These two organisational culture groups are complete opposites and a dramatic shift has taken place in organisations’ culture since 1999.

Various organisational factors were examined to establish their bearing on the deployment of systems development methodologies. The results revealed that IS department size has a negative impact on the developed system and the development process. It is also negatively related to the support aspects of methodology deployment. Maturity has a positive effect on the support for organisational alignment and horizontal use. The time dedicated to the development of new applications has a negative impact on the developed system as well as the development process. Developmental culture is positively related to the deployment of systems development methodologies.

At the individual level, it was found that relative advantage, ease of use, compatibility, demonstrability, trialability and developer support all have a positive impact on the individual deployment of systems development methodologies. Uncertainty, experience in systems development and voluntariness have a negative effect on different aspects of individual deployment.

OPSOMMING

Die doel van hierdie studie is om die gebruik en effektiwiteit van stelselontwikkelingsmetodologieë in die praktyk te ondersoek. Stelselontwikkelingsmetodologieë is gedefinieer as die kombinasie van stelselontwikkelingsbenaderings, prosesmodelle, metodes en tegnieke. Die term “ontplooing” verwys na die gebruik van stelselontwikkelingsmetodologieë, die ondersteuning wat hulle verskaf en die invloed wat hulle op die ontwikkelde stelsel en die ontwikkelingsproses het.

Die algemene doel van hierdie studie was om te ondersoek of daar ‘n verandering in die gebruik van stelselontwikkelingsmetodologieë sedert Huisman (1999) se studie plaasgevind het. ‘n Verdere doel was om te ondersoek of die faktore wat die gebruik en effektiwiteit van stelselontwikkelingsmetodologieë beïnvloed, wel verander het sedert 1999.

Navorsing was gedoen om die huidige toestand van die ontplooing van stelselontwikkelingsmetodologieë in Suid Afrika te ondersoek. Die resultate dui aan dat die ontwikkelingsomgewing uit verskeie ontwikkelingsplatforms, bedryfstelsels, en programmeringstale bestaan. Stelselontwikkelingstegnieke word meestal in die behoefteontleding en toetsing fases gebruik. Die algemene gebruik van stelselontwikkelingstegnieke was baie laag.

Die verwantskap tussen die volwassenheid van die sagteware ontwikkelingsproses in organisasies en die ontplooing van stelselontwikkelingsmetodologieë was ook ondersoek. Organisasies volg nie streng die volwassenheidsvlakke van die CMM nie en daar is ‘n afwyking van die voorgestelde vlakke. Die CMM vlak 3 aktiwiteite het die meeste impak gehad op die ontplooing van stelselontwikkelingsmetodologieë.

Die effek van ‘n organisasie se kultuur op die ontplooing van stelselontwikkelingsmetodologieë was nog ‘n area wat ondersoek is. In die algemeen het die organisasies swak waardes gehad vir al vier van die kultuur groepe soos gedefinieer deur die “Competing Values Framework”. Die kultuur groep wat die sterkste verwantskap met die ontplooing van stelselontwikkelingsmetodologieë gehad het was die ontwikkelings kultuurgroep, wat op desentralisasie, differensiasie, en buigbaarheid klem lê. Hierdie resultaat is betekenisvol aangesien Huisman (1999) gevind het dat die hiërargiese kultuurgroep die sterkste verwantskap met die ontplooing van stelselontwikkelingsmetodologieë gehad het. Hierdie twee kultuur groepe is die teenoorgestelde van mekaar en dié resultaat wys dat ‘n dramatiese verandering plaasgevind het in organisasie kultuur.

'n Aantal faktore is ondersoek om te bepaal of dit die ontplooiing van stelselontwikkelingsmetodologieë op organisasie vlak beïnvloed. Die resultate wys dat die grootte van die IS-departement 'n negatiewe impak op die ontwikkelde stelsel en die ontwikkelings proses het. Dit hou ook negatief verband met die ondesteinings aspekte van stelselontwikkelingsmetodologie ontplooiing. 'n Organisasie se volwassenheid, in terme van sy ontwikkelingsproses, hou positief verband met die organisasie se afstemming en horisontale gebruik. Die tyd wat 'n IS-departement aan die ontwikkeling van nuwe sagteware bestee het 'n negatiewe impak op die ontwikkelde stelsel en die ontwikkelings proses. Die ontwikkelings kultuurgroep hou positief verband met die ontplooiing van stelselontwikkelingsmetodologieë.

Verder is gevind dat relatiewe voordeel, kompleksiteit, versoenbaarheid, die moontlikheid om te eksperimenteer met stelselontwikkelingsmetodologieë, gebruik op 'n toetsbasis, en ontwikkelaar ondersteuning positief verband hou met die individuele ontplooiing van stelselontwikkelingsmetodologieë. Onsekerheid, ervaring in stelsels ontwikkeling en vrywillige gebruik het 'n negatiewe impak op die verskillende faktore van individuele stelselontwikkelingsmetodologie ontplooiing.

CONTENTS

CHAPTER 1

RESEARCH PROBLEM

1.1	Introduction	1
1.2	Research Problem	4
1.3	Research Approach	5
1.4	Expected Contributions	6
1.5	Outline of The Study	6

CHAPTER 2

LITERATURE STUDY

2.1	Introduction	8
2.2	Definition of a Systems Development Methodology	9
2.3	Systems Development Methodology Use	11
2.4	Systems Development Methodology Effectiveness	13
2.5	Software Failures	15
2.6	Systems Development Methodologies: An Historical Perspective	17
2.6.1	The Pre-Methodology Era	17
2.6.2	The Early-Methodology Era	18
2.6.3	The Methodology Era	21
2.6.4	The Post-Methodology Era	29
2.7	The Dawn of a New Era: Current and Future Trends of Systems Development Methodologies	30
2.7.1	Component-Based Software Development	30
2.7.2	Web Application Development	34
2.7.3	Extreme Programming	36
2.8	Discussion and Final Comments	39

CHAPTER 3

CONCEPTUAL RESEARCH MODEL AND RESEARCH METHOD

3.1	Introduction	40
3.2	Conceptual Research Model	40

3.2.1	The Innovation: Systems Development Methodologies	40
3.2.2	Innovation Diffusion Process	41
3.2.3	Adopting Unit	41
3.3	Research Method	44
3.3.1	Development of the Questionnaire.....	44
3.3.2	Distribution of the Questionnaire.....	46
3.4	Discussion and Final Comments	50

CHAPTER 4

RESULTS OF DESCRIPTIVE STATISTICAL ANALYSES

4.1	Introduction	52
4.2	Descriptive Statistics	52
4.2.1	Profile of Organisations And Developers.....	52
4.2.2	Characteristics of The Development Environment in IS Organisations	54
4.2.4	Characteristics of Systems Development Methodology Use	59
4.3	Discussion and Final Comments	64

CHAPTER 5

PERCEIVED MATURITY OF IS DEPARTMENTS AND THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

5.1	Introduction.....	65
5.2	The Maturity of IS Departments	66
5.3	The Deployment of Systems Development Methodologies	68
5.3.1	The Use of Systems Development Methodologies	69
5.3.2	Perceived Methodology Support.....	70
5.3.3	Perceived Impact of Systems Development Methodologies	70
5.4	Results.....	71
5.4.1	Distribution of IS Departments Across Maturity Levels.....	71
5.4.2	Regression Results	73
5.5	Discussion and Final Comments	76

CHAPTER 6

ORGANISATIONAL CULTURE AND THE DEPLOYMENT OF SYSTEMS

DEVELOPMENT METHODOLOGIES

6.1	Introduction	77
6.2	Organisational Culture, the Competing Values Framework and the Deployment of Systems Development Methodologies	78
6.2.1	Organisational Culture	78
6.2.2	The Competing Values Framework	79
6.3	The Deployment of Systems Development Methodologies	80
6.3.1	The Use of Systems Development Methodologies	80
6.3.2	Perceived Methodology Support.....	81
6.3.3	Perceived Impact of Systems Development Methodologies	81
6.4	Results.....	81
6.4.1	Cluster Analysis	81
6.4.2	Regression Results	83
6.5	Discussion and Final Comments	86

CHAPTER 7

THE ORGANISATIONAL DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

7.1	Introduction	88
7.2	Organisational Characteristics and the Deployment of Systems Development Methodologies	88
7.2.1	Organisational Characteristics	89
7.2.2	Task Characteristics	91
7.2.3	Deployment of Systems Development Methodologies	91
7.3	Results.....	92
7.3	Discussion and Final Comments	97

CHAPTER 8

THE INDIVIDUAL DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

8.1	Introduction	100
8.2	Individual Characteristics and the Deployment of Systems Development Methodologies	100
8.2.1	Innovation Characteristics	101
8.2.2	Individual Characteristics	102
8.2.3	Task Characteristics	103
8.2.4	Organisational Characteristics	103
8.2.6	Innovation Diffusion Process Characteristics.....	103
8.2.7	Deployment of Systems Development Methodologies	104
8.3	Results.....	105
8.4	Discussion and Final Comments	115

CHAPTER 9

SUMMARY AND FINAL CONCLUSIONS

9.1	Introduction	118
9.2	Contributions of the Study	118
9.2.1	The Current State of Systems Development Methodology Use In South Africa.....	119
9.2.2	Perceived Maturity of IS Departments and the Deployment of Systems Development Methodologies	120
9.2.3	Organisational Culture and the Deployment of Systems Development Methodologies.....	121
9.2.4	Organisational Deployment of Systems Development Methodologies	122
9.2.5	Individual Deployment of Systems Development Methodologies	123
9.3	Limitations and Future Research.....	124
9.3.1	Limitations of the Study	124
9.3.2	Future Research	124

BIBLIOGRAPHY

126

CHAPTER 1

RESEARCH PROBLEM

1.1 INTRODUCTION

An historical perspective on systems development methodologies shows that they came into being to address deficiencies in existing techniques and to introduce some rigour into the software design process (Avison and Fitzgerald, 2003). The systems (or software) development life cycle (SDLC) was one of the first such methodologies to be introduced into the academic community and it subsequently became the systems development methodology for the 1970s (Lee, 1987). It was developed in an attempt to produce information systems that were delivered on time, within budget and closer to the requirements of the user (Walters *et al.*, 1994), analogous to the goals of present day SDMs.

Methodologies and their uses have evolved with time, and Avison and Fitzgerald (2003) have created distinct categories that describe this evolution through the years, namely pre-methodology era, early methodology era, methodology era, and post methodology era. Prior to the existence of software development methodologies, computer applications were developed on an ad-hoc basis and developers were trained purely in computer technology with little or no understanding of the business or organisational contexts in which the systems were to be implemented. The period in which this type of ad-hoc development took place can be seen as a *pre-methodology era*.

The introduction of the Systems Development Lifecycle commenced the *early methodology era*. Although the SDLC contributed to making the systems development process more thorough, it was not the panacea to all systems development problems and it also had fundamental problems of its own (Walters *et al.*, 1994; Avison and Fitzgerald, 2003). As a result of these problems, a number of lessons were learnt by information systems developers and users.

First, the use of a life cycle consisting of documentation, control, and training procedures is not enough to ensure a successful system. Second, the development of an information system should take an organization-wide view, as failings may be due to the narrowness with which the analyst perceives the information system (Walters *et al.*, 1994; Avison and Fitzgerald 2003).

Due to the failings of the SDLC, a number of newer methodologies emerged in the *methodology era* with the purpose of producing better end-products, improving the development process and creating a standardised process for software development within an organisation. Seven broad themes or approaches of methodologies emerged (Avison and Fitzgerald 2003):

- **Structured** – Concepts of structured programming were used and techniques such as data-flow diagramming enabled the top-down analysis and representation of complex processes.
- **Data Oriented** – Emphasis was placed on the understanding of data as the key element and the most prominent technique used was entity-relationship modelling.
- **Prototyping** – An approximation of the system was first built so that users could visualise and respond to it prior to its physical implementation.
- **Object Orientated** – The identification of objects, attributes and classes helped provide the theoretical benefits of inheritance and reuse.
- **Participative** – The crucial feature was involvement of users and stakeholders.
- **Strategic** – The emphasis was the planning of information systems and the development of an information systems strategy to support and enable the overall objectives of the business.
- **Systems** – The complexities of human activity systems were addressed by adopting a holistic view far beyond a system's single-application boundaries.

The use of SDMs is encouraged and recommended for various reasons (Fitzgerald, 1996). Firstly, organisations are under pressure to maintain their competitiveness by adhering to international standards as set by the International Standards Organisation (ISO).

Secondly, the Software Capability Evaluation Program of the Software Engineering Institute assesses the capability of organisations to produce high-quality software on time. This program emphasises adherence to formalised development procedures (Humphrey, 1989). Thirdly, some governments are enforcing systems development standards (Fitzgerald *et al*, 2002). By doing so, they force organisations that deliver software to them, to use the appropriate systems development methodology.

The *post methodology era* has seen many companies overwhelmed with the multitude of SDMs that they can choose from, and often, unsure of whether they need to use a methodology (Avison and Fitzgerald, 2003). Although there are many reports of companies that have implemented a

methodology, and in so doing, successfully completed a software development project, there are increasing accounts of companies that encountered failure even though they had implemented an SDM (Ewusi-Mensah, 1997; Fitzgerald *et al*, 2002). As a result, organisations that have attempted using a commercially available methodology have either, developed and followed their own custom methodology, adapted an existing methodology to suit their purposes or, abandoned the use of methodologies altogether (Nandhakumar and Avison, 1999; Avison and Fitzgerald, 2003). It is the latter decision that is questioning the use and effectiveness of systems development methodologies.

Furthermore, ineffective SDMs are an important risk category that is of major concern to software development managers (Lyytinen *et al.*, 1996; Lyytinen *et al* 1998). The failure of many information systems development projects has also questioned the relevance of systems development methodologies. Why are projects abandoned halfway, at a great cost to the organisation in terms of time and money, even though a systems development methodology was followed rigorously? (Ewusi-Mensah, 1997)

Failed projects cost an organisation a lot of time and resources, so software development managers need to carefully consider the risks involved when selecting a new SDM. They also need to ensure that the chosen methodology applies to the processes followed in the entire organisation and not only in the software development department (Walters *et al.*, 1994).

Consequently, the problem of evaluating the effectiveness of SDMs becomes one of theoretical and practical importance (Tolvanen, 1995; Kelly and Rossi, 1998; Siau and Rossi, 1998). Systems development methodologies have a critical strategic impact on organizational performance in that they are the means for constructing, adapting and renewing the IT infrastructure of organisations (Beynon-Davies and Williams, 2003).

It is vital to determine the impact that SDMs have on the development of systems to better understand the software crisis that has distressed this profession. The applications of computer software are constantly escalating. Although the fundamentals of SDMs from years past might still apply, newer approaches to software development can perhaps aid in alleviating some of the problems associated with the software crisis that past methods could not resolve.

Evaluating systems development methodologies is thus a persistent and intractable problem and accordingly, warrants more research (Tolvanen *et al.*, 1996). This study will try to determine whether systems development methodologies are still considered to be useful and effective in their implementation. This has become especially pertinent since it is claimed that the software development industry is experiencing a paradigm shift into the *post-methodology* era (Avison and Fitzgerald, 2003).

1.2 RESEARCH PROBLEM

This study focuses on the use and effectiveness of systems development methodologies (SDMs) in South Africa and ensues upon the research conducted by Huisman in “The Deployment of Systems Development Methodologies: A South African Experience” (1999). The term deployment encompasses the use of systems development methodologies, the perceived support provided by them, and their impact on both the developed system and the development process. One of the objectives of Huisman’s study was an investigation of the status of systems development methodology deployment in South Africa.

The primary purpose of this study is to determine whether there has been a change in the frequency and intensity of use of SDMs since 1999. In the case where a methodology was used, an attempt will be made to gauge the effectiveness of this methodology.

The types of systems being developed at present differ considerably from those that were developed five years ago. For example, web applications have become popular out of necessity due to the increased dependence of human beings on the Internet (Avison and Fitzgerald, 2003; Glass, 2003). Another change in developmental practice has seen the speed of application development escalate due to the demand for computer software systems (Avison and Fitzgerald, 2003).

The general purpose of this research is to investigate whether there has been a change in the use of SDMs in South Africa since (Huisman, 1999) and also whether the factors that influence the use and effectiveness of SDMs have changed or remained the same as put forth in 1999.

Much of the existing empirical research on systems development methodologies has focused on specific methods and techniques, rather than encompassing the general idea of a methodology, where a methodology is interpreted as a totality of systems development approaches, process models, methods and techniques used in an organisation (Beynon-Davies, 1998; Avison and Fitzgerald, 2003). This study encompasses the general idea of a methodology and does not focus on any specific methodology.

The study is based in South Africa since even less is known about the use and effectiveness of systems development methodologies here. In conducting the literature study, only two previous studies reporting on the use of systems development methodologies in South Africa were identified prior to the study of Huisman (1999), i.e. Erlank *et al.* (1991) and Addison and Hamersma (1996). This does not imply that the findings of this research will only be applicable to South Africa, instead, the expectation is that the analyses can be done at a more theoretical and abstract level, so that the findings are also of import elsewhere.

1.3 RESEARCH APPROACH

The investigation into the use and effectiveness of systems development methodologies can be based on a large number of research approaches which are dependent on the following:

- *Research target:* The focal point of this study is systems development methodologies as a general technology.
- *Research method:* An electronic survey focusing on systems developers and managers at the individual level and IS departments at the organisational level.
- *Theoretical orientation:* The conceptual research model developed and used by Huisman (1999) which was primarily based on the innovation diffusion theory, and supplemented by other theories.
- *Adopting unit and stakeholder groups:* Individual (IS developers and managers) and organisational (IS department).

1.4 **EXPECTED CONTRIBUTIONS**

This study investigates whether there has been a change in the use of systems development methodologies since Huisman's study in 1999. More concretely, it aims to answer the following research questions:

- Has the deployment of SDM in South Africa changed since 1999?
- Has a change occurred in the relationship between the maturity of an IS department and the deployment of SDM since 1999?
- Has a change occurred in the relationship between organisational culture and the deployment of SDM since 1999?
- Has a change occurred in the factors that explain the organisational deployment of SDM since 1999?
- Has a change occurred in the factors that explain the individual deployment of SDM since 1999?

1.5 **OUTLINE OF THE STUDY**

This dissertation is organised as follows:

Chapter 1: Research Problem

This chapter gives a brief introduction to the dissertation and defines the research problem of the study.

Chapter 2: Literature Study

An historical overview of systems development methodologies is given and a formal definition of a systems development methodology is presented within this chapter.

Chapter 3: Conceptual Research Model and Research Method

The conceptual research model on which the dissertation is based as well as the research method used in obtaining the data is discussed in this chapter. The response rate received to the survey is also presented in this chapter.

Chapter 4: Results of Descriptive Statistical Analyses

The results of the statistical analyses that were carried out on the data gathered from the survey are presented in this chapter.

Chapter 5: Perceived Maturity of IS Departments and the Deployment of Systems Development Methodologies

The relationship between the perceived maturity of an IS department and the deployment of systems development methodologies is studied in this chapter.

Chapter 6: Organisational Culture and the Deployment of Systems Development Methodologies

An investigation into the relationship between organisational culture and the deployment of systems development methodologies is conducted within this chapter.

Chapter 7: The Organisational Deployment of Systems Development Methodologies

The factors influencing the organisational deployment of methodologies are studied.

Chapter 8: The Individual Deployment of Systems Development Methodologies

The factors influencing the individual deployment of methodologies are studied.

Chapter 9: Summary and Final Conclusions

The final chapter offers a summary of the results and theorises on possible future research.

CHAPTER 2

LITERATURE STUDY

2.1 INTRODUCTION

The systems development methodology (SDM) domain is a vast and ever-evolving area of information systems development. From as early as the 1960's, it was realised that a methodology was needed in order to decrease the risk of failure of software development projects (Avison and Fitzgerald, 2003; Walters *et al.*, 1994).

As the scope and complexity of these projects increased, a need for new and improved SDMs emerged. Organisations now needed to develop new systems at an increasingly faster rate, and with a lower proportion of failures, in order to remain competitive in the corporate world. This led to the development of a new generation of systems development methodologies that introduced innovative and modern approaches to the development of software systems. These included methodologies such as Rational Rose, Information Engineering and Prototyping methodologies.

In recent times however, there appears to be a deviation from the use of such methodologies. This is characteristic of the post-methodology era as described by Avison and Fitzgerald (2003). Many organisations have instead opted to modify existing methodologies to suit their purposes as an alternative to using them in their standardised form (Beynon-Davies and Williams, 2003).

This chapter will discuss the literature survey conducted and present the findings obtained in the literature. A formal definition of a systems development methodology will also be provided together with a discussion of the transformations that SDMs have undergone since 1999. This discussion will examine a few of the newer methodologies that have emerged in recent years and the perceived effectiveness of their use.

2.2 DEFINITION OF A SYSTEMS DEVELOPMENT METHODOLOGY

Defining the term “Systems Development Methodology” is not a clear-cut task. Definitions range from the simple to the complex and there is no universally accepted, rigorous and concise definition of an information systems development methodology (Avison and Fitzgerald, 1995; Wynekoop and Russo, 1997; Iivari *et al.*, 1999).

Some definitions of systems development methodology/method include:

- A systems development methodology is a generalised set of methods and procedures used in projects (Veryard, 1985).
- Systems development methods incorporate both methodologies and process models (Wynekoop and Russo, 1993)
- A systems development methodology is a collection of procedures, techniques, tools, and documentation aids that will help systems developers in their efforts to implement a new information system. A methodology will consist of phases, themselves consisting of sub-phases, which will guide the systems developers in their choice of the techniques that might be appropriate at each stage of the project. It also helps them plan, manage, control and evaluate information systems projects. The methodology must also be based on some philosophical view (Avison and Fitzgerald, 1995).
- A method is an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities, with corresponding development products (Brinkkemper, 1996)
- A systems development methodology is a systematic procedure for completing either a system or one of several stages of the systems development life cycle. It consists of goals, principles and specific methods and tools, which are selected on the basis of an underlying rationale or system development philosophy (Iivari *et al.*, 1999).

A certain amount of ambiguity remains in the various definitions listed. The distinction between a *method* and a *methodology* is not clear. Some researchers argue that the term “methodology” is inappropriate since it literally means a “science of methods” (Baskerville *et al.*, 1992; Schach, 1997). Others argue that the term can be applied interchangeably (Connors, 1992; Saeki, 1998).

Despite this, the following four elements were identified in the various definitions of a systems development method/methodology (Huisman, 1999):

- The systems development method/methodology itself.
- A systems development method/methodology is based on some philosophical view or approach
- A systems development method/methodology includes a set of techniques.
- A systems development method/methodology follows a process model.

Considering the four elements identified above, the following definition of a systems development methodology is formulated (Huisman, 1999):

- A systems development approach(es):
This represents the philosophical view on which the methodology is built. It is the set of goals, guiding principles and beliefs, fundamental concepts and principles of the systems development process that drive interpretations and actions in systems development (Iivari *et al.*, 1998; Iivari *et al.*, 1999). Examples of systems development approaches are the structured approach, object-oriented approach, information modelling, etc.
- A systems development process model(s):
Wynekoop and Russo (1993) define a process model as a representation of the sequences of stages through which a system evolves. Some examples of process models are the linear lifecycle model and the spiral model.
- A systems development method(s):
A method is a systematic approach to conducting at least one complete phase of systems development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of systems development and the target system (Wynekoop and Russo, 1993). Examples include OMT, IE, etc.
- A systems development technique(s):
Systems development techniques can be defined as a procedure, possibly with a prescribed notation, to perform a development activity (Brinkkemper, 1996) for example entity relationship diagrams.

2.3 SYSTEMS DEVELOPMENT METHODOLOGY USE

The use of methodologies for the development of systems has been a point of interest in the academic world for a long period of time. The major point of contention has been whether available methodologies do in fact assist in the development process, i.e. whether they are deemed to be effective in their use or not. There is however an inadequate amount of research available to corroborate that fact, and not much is known about the actual use of methodologies in practice.

A summary of empirical studies by various researchers on the use of systems development methodologies appears in Huisman (1999). This empirical proof is contradictory and inconsistent. Chatzoglou and Macaullay (1996) found that 47% of respondents never used a methodology, while Hardy *et al.* (1995) found the number to be 18%. The conflicting use of the term method, methodology, technique and process model has also led to variation in results obtained in studies conducted by Eva and Guilford (1996) and Wynekoop and Russo (1993).

Notwithstanding these inconsistencies, a cumulative view of past research does indicate a lack of use of systems development methodologies in organisations. The level of use of techniques and methodologies in use by IS professionals in the USA was studied by Palvia and Nosek (1993).

Their comment upon analysis of the findings was the following:

“One startling and somewhat disturbing observation is that many methods are used very little”.

A study by Fitzgerald (1996) found that an educated decision not to use systems development methodologies is made by certain practitioners because they have been made aware of the restrictions of the methodologies.

Other research shows that systems development methodologies are adapted to suit the purposes of the organisation. The SDM is seldom used exactly as specified by organisations. They instead choose to customise the systems development methodology according to the requirements of the projects, or apply only certain aspects of the methodology (Van de Velde, 1992; Dietrich *et al.*, 1997; Hughes, 1998). As a consequence, newer versions of existing methodologies often recommend some sort of adaptation or tailoring. There is however very little guidance available to developers as to what steps to modify or omit and as a result much is left to the intuition of individual developers.

An attempt to find similar, more recent empirical research regarding the general use of systems development methodologies, proved to be unsuccessful.

The face of software development has changed considerably in recent years and today, information systems development is often about deploying complete solutions which may be purchased ready-made off the shelf. Alternatively, configuring ready-made components into systems, or using high-level computer programming tools to produce software products, has also become a popular option. Notwithstanding this, it is still up to the developer to investigate, analyse and design the system's purpose, function, structure, appearance, behaviour, cost and efficiency. Considering this fact, it seems apparent that the development and use of information systems will not become obsolete (Fitzgerald *et al.*, 2002).

Research on the use of systems development methodologies in South Africa is sparse and only three previous empirical studies were identified. Erlank *et al.* (1996) investigated the incidence of usage of systems development tools and techniques in South Africa, and systems development methodology use was covered very briefly. Their study revealed that fifty-two percent of organisations used a systems development methodology.

Addison and Hamersma (1996) conducted a study to determine the critical success factors for implementing upper CASE technology. They found a mixture of commercial and in-house developed systems development methodologies was present in organisations but no perceivable trend was identified. The focus of this study was on upper CASE technology, and systems development methodology use was studied as one of its critical success factors.

Huisman (1999) conducted a study which investigated the deployment of systems development methodologies in South Africa. Although this study did reveal significant findings related to the use of SDMs, it is possible that there has been a change in the use of systems development methodologies since then. Considering this, there is still a necessity for additional research on this topic in South Africa.

2.4 SYSTEMS DEVELOPMENT METHODOLOGY EFFECTIVENESS

Systems development methodologies have been endorsed as being capable of rendering the development process to be more effective, secure, predictable and easier to control (Fitzgerald *et al.*, 2002). In addition to this, there are several other advantages to using methodologies (Avison and Fitzgerald, 2003; Ambler, 2001).

- Methodologies assist in the specification of accurate requirements for an information system.
- A systematic method of development is provided so that progress can be monitored effectively.
- Methodologies ensure that an information system is completed within an appropriate time frame and at a cost within the project's budget.
- When a methodology is used, the system produced has adequate documentation and is thus easier to maintain.
- The use of a methodology also gives an early indication of the changes that need to be made during the development process.
- Since the development of the information system is managed by a standardised process which promotes reuse and consistency, a better quality end-product is created.
- A methodology improves the organisation's operations and support efforts by managing change and facilitating the smooth transition of software into operations and support.

Existing research however, suggests that these methodologies have not always been used, and if used, not always in the way they were intended to be used. An investigation into existing research on SDM effectiveness up to 1999 yields some mixed results (Huisman, 1999). Reasons for this include:

- To facilitate the measurement of systems development methodology effectiveness, there is a requirement for some sort of criteria by which its efficacy can be assessed. There are however no such criteria available for use. Wynekoop and Russo (1997) suggested the following criteria: user satisfaction with the product, developer satisfaction with the process, design complexity, system maintainability, system quality, and developer productivity.

- A particular systems development methodology may not be applicable to all organisations. Studies reveal that many organisations use a number of different systems development methodologies (Punter and Lemmen, 1996; Eva and Guilford, 1996). Research has also shown that developers rarely follow the steps as prescribed in a methodology. There is, as a result, an adaptation of the methodology or a blending of different methodologies to fit the needs of the organisation (Fitzgerald *et al.*, 2002).
- The environment in which systems development methodologies are utilised also needs to be considered. Information systems are created, designed and produced for a specific situation or at least a specific type of situation. A myriad of factors can influence the success or failure of a systems development methodology, e.g. organisational, individual and environmental factors.

Empirical research conducted by Fitzgerald *et al.* (2002) shows that the use of formalised SDMs was significantly higher in larger organisations (more than 1000 employees) and larger IS departments (more than 20 personnel).

The popularity of agile methodologies is steadily increasing since their introduction into the software development arena in the 1990's (Good, 2003). An empirical survey conducted by Reifer (2002) reflects the benefits that organisations experienced from using agile methodologies. The results are shown below.

- A gain of fifteen to twenty percent in productivity was reported by the organisations that participated in the survey.
- A cost reduction of between five and seven percent was recounted.
- The time-to-market was reduced by up to fifty percent compared to previous projects.
- An improvement in the quality of the developed system was another benefit that sixteen percent of the organisations experienced.

Another study conducted by Shine Technologies (2003) yielded the following results:

- Eighty-eight percent of organisations cited an improvement in productivity.
- Eighty-four percent of the organisations indicated an improvement in the quality of software production.
- Forty-nine percent of the organisations in the study stated that they experienced a significant reduction in costs when using agile methodologies.

- Eighty-three percent also stated that business satisfaction with the software, and the process followed to develop it, was significantly higher.

Although the empirical results reported on are focused on agile methodologies, they do clearly indicate that organisations find the implementation of a software development methodology to be effective. However, there are many factors that need to be taken into account in the planning of a development project and an awareness of these factors must be in place to assist with the choice of a methodology.

2.5 SOFTWARE FAILURES

The use of software methodologies is said to decrease the risk of failure of an IS project (Hull *et al*, 2002; Avison and Fitzgerald, 2003). However, a study in the *PC Week* magazine (1995), reported that 31% of new IS projects are cancelled before completion at an estimated combined cost of \$81 billion. Additionally, 53% of the projects completed are 189% over budget at an extra cost of \$59 billion. These findings would cause even the most ardent methodology users to stop and consider whether the use of a methodology would assist in preventing these failures.

The reasons for software failures are various, but the following seem to be mentioned most frequently in the aftermath of a software project failure (Hull *et al*, 2002; Ewusi-Mensah, 1997):

- Ad-hoc requirements management
- Ambiguous and imprecise communication
- Brittle architectures
- Overwhelming complexity
- Undetected inconsistencies in requirements, designs, and implementations
- Insufficient testing
- Subjective project status assessment
- Failure to attack risk
- Uncontrolled change propagation
- Insufficient automation

The reasons for software failures mentioned above are somewhat surprising since organisations that have used methodologies should not be having these problems. However, many organisations are renouncing the use of methodologies in the post-methodology era and this may cause the number of failed projects to increase.

The Rational Software Corporation (1999) suggests the following six best practices to combat software failures:

- ***Develop Software with an Iterative and Incremental Approach*** – the understanding of the problem domain and its solution is increased by developing iteratively. Successive refinements and to this solution are also made possible by the incremental growth of the particular solution.
- ***Manage Requirements*** – Requirements need to be elicited, organised and documented. Any changes to these requirements must be assessed and trade-offs should be tracked and documented. In this manner, inconsistencies can be detected and an objective assessment of functionality and performance is made possible.
- ***Use Component-Based Methodologies*** – Component-based methodologies are more resilient to change and provide increased support for reuse. Configuration management is also simplified when components are used.
- ***Visually Model Software*** – The design of the software system is represented unambiguously and complexity can be hidden or exposed as necessary.
- ***Verify Software Quality*** – Continuous testing during iterations makes it possible to uncover inconsistencies in requirements, designs, and implementations early when the cost is relatively low.
- ***Control Changes to Software*** – Often, developers in numerous teams are at different geographical locations and this necessitates the control of changes.

Software failures cost organisations money and resources and often cause the individuals within the organisation to feel unsure of themselves. It is thus of utmost importance that an effort is made to prevent software failures, which in turn will help in alleviating the software crisis that is currently being experienced.

2.6 SYSTEMS DEVELOPMENT METHODOLOGIES: AN HISTORICAL PERSPECTIVE

The emergence of systems development methodologies and their impact on the software industry will be discussed in this section. The different eras of software development methodology use (Avison and Fitzgerald, 2003; Fitzgerald *et al.*, 2002) will be discussed next. This is vital in order to gain some insight into the past of SDMs and hypothesise on what future trends may be.

2.6.1 The Pre-Methodology Era

The pre-methodology era began with computers being used primarily for scientific problem solving. The accuracy and efficiency of computers and their ability to perform mathematical calculations rapidly, made them pertinent in fields such as the calculation of missile trajectories, aerodynamics and seismic data analysis. The primary users of computers were accordingly scientific researchers, with a strong mathematical or engineering background.

During the early 1950's, computer use spread beyond that of scientific problem solving and into the area of business data processing. These early machines were responsible for accurately capturing organisations' business transactions. By 1960, this trend had quickly spread, and the business data processing application had overtaken its scientific counterpart.

Scientific computation and business data processing are two applications that are widely divergent. Business applications involve high input and output and the peripherals used at that time for this were time-consuming. This created a necessity for developers to write "good" programs that performed a task efficiently. Clear, well-documented and easily understood programs were rejected in favour of efficient programs.

Formal training for developers was unheard of and programming skills were often learnt solely through experience. The lack of standards in programming practice, combined with the individuality and creativity of novice programmers, all coupled with the necessity to write programs that were as concise and efficient as possible, led to the development of complex programs that were difficult to understand. This inevitably led to making program maintenance problematic.

Another problem that emerged in this era was that of determining requirements for a system. The scientific background of computer use contributed to the propagation of this problem since scientific researchers were accustomed to developing programs with themselves being the primary users. The business data processing environment saw this change since the systems developers were seldom the actual users of the system. This led to a communication gap between the developer and user which contributed to making the specification of systems requirements difficult. The result of this was that development projects were being delivered with the priority of remaining within budget and time constraints, which compromised the quality of the system. The term “software crisis” was coined as early as the 1960’s and it encompassed the problem of systems exceeding time and budget targets and failing to meet user requirements.

Computers provided unmistakeable functionality, but the problems that accompanied this functionality caused developers to begin formulating some systematic approaches to the development of software.

2.6.2 The Early-Methodology Era

During the early-methodology era, developers realised that systems development involved more than just program coding. There emerged an increased interest in the analysis and design phases of developing software.

The origins of early systems development methodologies lay in the field of industrial engineering from several decades in the past. Process flow charts were used to depict the flow of materials and the points at which transformation occurred. These were supplemented by form flow charts which utilised special symbols for files and forms.

Flow charts provided too little information and required substantial supplementation with narrative annotation. A need thus arose for a more comprehensive way to analyse and design software.

An example of a methodology that provided more comprehensive analysis was ADS (Accurately Defined Systems). ADS was primarily focused on the requirements specification phase of systems development and used a series of interrelated forms to define input, output, computation, historical information and logic definition.

Other methodologies focused on an earlier phase of systems development, starting from the study of the organisation and its information needs. For example, SOP (Study Organisation Plan) which was developed by IBM, was concerned with gathering a varied and large set of data to analyse the information needs of the entire organisation.

TAG (Time Automated Grid) was a methodology which placed much more emphasis on systems for the entire organisation rather than individual departments. Additionally, the computer was used to assist in the analysis process. Other methodologies which had an element of automation included ISDOS (Information Systems Design and Optimisation System). ISDOS used a problem statement analyser (PSA) to analyse problem statement language (PSL) and produce comprehensive data and function dictionaries, an analysis of data relationships, and eventually, a machine-readable problem statement which could be used for direct physical system design (Teichrow and Sayani, 1971).

Although some of the early methodologies did make breakthroughs in systems design and analysis, they were not widely used mainly because of their complex nature and the lack of support provided. However, many of the basic principles of these methodologies have been incorporated into later methodologies.

For example, identifying the organisations information needs; the concept of starting with the output the system must produce and using this to derive the necessary input; defining an underpinning database; graphical representation of system structures; capture of requirements in a formal specification – all of these principles have been incorporated into methodologies such as Information Engineering, Jackson Systems Development and the Structured Approach.

- **The Systems Development Lifecycle**

The use of the systems development lifecycle (SDLC) during the early-methodology era exemplified the fundamental characteristic of this era i.e. building computer-based applications by focusing on the identification of phases and stages.

The origins of the SDLC can be traced back to general systems theory of the 1930's. Its introduction to the systems development arena was motivated by its use in operations research in the 1950's where the principles of the lifecycle were applied as an approach to problem-solving.

As the popularity of computing increased, researchers familiar with the systems approach proposed lifecycle models for system development. An example of such a model was proposed by Canning (1956) and was comprised of the following four phases:

- Systems study (includes requirements gathering)
- Preliminary design
- Detailed design
- Programming/acquisition/installation

The lifecycle model was further modified and elaborated on during the 1960's. Concentration was steered towards the verification of deliverables from each phase. This ensured that each stage was satisfactorily completed before the next one began. The waterfall model, as presented by Royce (1970), thus came into being.

There are a number of fundamental characteristics which are inherent in the SDLC concept (Fitzgerald *et al.*, 2002): The lifecycle consists of *discrete* stages, each of which has distinct activities and a specific end-product which serves as input to the next stage. At the end of any stage, the decision to proceed is only valid until the completion of the next stage which creates a *stage-limited commitment*. At the completion of each stage there is a *signoff of interim end-products* where stage deliverables are verified and feedback is provided to ensure each stage has been completed satisfactorily. There may also be several *in-stage reviews* within each stage with the purpose of trying to identify errors at the earliest possible time.

The nature of the software crisis caused the SDLC to be viewed primarily as a prescriptive device to control projects i.e. it acted as a checklist to govern when and how different activities should be performed and what resources would be required. However, when considering it from a practical perspective, the SDLC offered much to promote its use. Firstly, it ensured a top-down approach, from a broad overall view to technical detail, thereby taking a “divide and conquer” approach to dealing with complexity. Secondly, it avoided premature coding since it had a logical sequence of steps to follow. Lastly, design preceded coding, which in turn, was preceded by requirements. This ensured that the system was planned adequately before coding took place.

Although the SDLC benefited systems development to a large extent, it still had its limitations, including the way in which it was used (Avison and Fitzgerald, 2003). The most noteworthy problems were:

- Failure to meet the real needs of the business due to concentration on technological efficiency improvements at the operational level of the organisation.
- Overly conservative systems design due to emphasis on the existing system as a basis for the new system.
- Instability caused by the lack of flexibility in modelling processes to accommodate for changing business needs and markets.
- Inflexibility due to the output-driven orientation of design processes, thus making changes in design difficult and costly.
- User dissatisfaction due to problems with computer-oriented documentation and users' inability to "see" the systems before it is operational.
- Application backlog caused by increased maintenance workload as attempts are made to change the system in order to reflect changing user needs.

Notwithstanding these shortfalls, many principles of the systems development lifecycle were expanded on and refined in the next era of systems development.

2.6.3 The Methodology Era

The SDLC gave software developers very little direct benefit, but during the methodology era, many of the concepts of the systems development lifecycle were extrapolated on to form new methodologies that attempted to improve upon their predecessors.

Four major flavours of methodologies emerged during this era: The structured approach, data driven approach, object orientation and prototyping. Each of these will be discussed next.

- **The Structured Approach**

The structured approach is said to be the most widely-used development methodology in practice (Russo *et al.*, 1995). Based on the waterfall lifecycle; the specification, design and coding phases inherent in the waterfall model, map well to the structured analysis, design and programming activities of the structured approach.

Structured programming attempted to eliminate the individualistic programming practices of the pre- and early-methodology eras. The principle behind this was that programmers be restricted to using only three constructs in their programs i.e. sequence, selection and iteration. Also, it was recommended that unconditional branching is avoided (Yourdon, 1979).

Structured programming on its own did little other than to reveal inadequacies further up in the design process (Fitzgerald *et al.*, 2002). The focus was thus shifted and broadened to consider systems design. Structured design incorporated practices such as a top-down, functional decomposition approach to program design, and modularisation of code. Functional decomposition is the process of progressively subdividing primary functions into sub-functions, until a primitive level from which individual program modules can be written is reached. It is a prime example of the “divide and conquer” concept, which breaks complex tasks into manageable sub-tasks.

Structured design introduced some valuable concepts into the systems development arena. One of the central principles of structured design is information hiding, which asserts that each module should hide exactly one design decision, and reveal as little as possible about its inner working or the data that it uses (Parnas, 1972). Cohesion and coupling are two more concepts vital to the correct approach to modularisation. Cohesion refers to the extent to which an individual module is self-contained and performs a single well-defined function. Coupling in contrast, refers to the interdependence of modules, that is, the extent to which changes to any particular module affect other modules. Other techniques that form part of structured design are structure charts – a graphical representation of the system structure, mini specifications – a type of pseudo code which allows a terse and unambiguous representation of module logic using just the permissible structured programming constructs, and transform analysis and transaction analysis – different strategies which can be used to decompose functions (DeMarco, 1978).

Structured programming and design ensured that the system would be well-designed and well-structured, but didn't assist in analysing the problem domain and solving the basic problem that the system set out to address. The structured analysis approach arose to deal with this issue. Its principal tools and techniques included Data Flow Diagrams (DFDs), Entity Relationship Diagrams (ERDs) and the Data Dictionary (DeMarco, 1978).

The structured approach was one of the “silver bullets” in the software industry. It was the most widely used methodology in North America and Europe but it too had a fair share of criticisms and weaknesses.

The representational shift from the analysis stage to the design stage is a major weakness in the structured approach. The creation of the hierarchical structure charts from the data flow diagrams is poorly defined which causes the design to be loosely coupled to the results of the analysis (Colter, 1982), and consequently, there is discontinuity in the transition from the analysis to the design phase (Coad and Yourdon, 1991). The logical-physical separation of the structured approach has also been criticised since logically, a solution could be proposed, but it could turn out to be physically impossible to implement (Henson and Hughes, 1991).

An empirical survey of software development practitioners by Sumner and Sitek (1986) found that although the benefits of the structured approach were generally acknowledged by practitioners, they often did not use it in practice. The two main reasons cited by respondents for lack of use in this survey were that the methodology was too time-consuming, and its lack of acceptance amongst end-users.

Another survey by Bergland (1981) pointed out that structured techniques such as programmer teams, design and code reviews, and walk-throughs were readily applied. However, other more fundamental concepts such as the necessity of having well-structured code, were not readily implemented. This finding suggests that the structured approach may not, in practice, be implemented in its entirety as a development methodology, but individual concepts and techniques are probably widely used.

The structured approach has been criticised for disregarding softer issues such as differing interests and power amongst stakeholders of the system. It has also been criticised for the failure to recognise the need for negotiating and establishing consensus among users about the purpose of the new system (Bansler and Bodker, 1993)

The structured approach may have had a number of shortfalls, but it has left a legacy of three key principles (Gane, 1991):

- Emphasis on the importance of making programs maintainable, which results from the increased clarity of having well-structured programs that use standard constructs.

- Facilitation of the changeability of programs due to adherence to information hiding, cohesion, and coupling guidelines.
- Improved visibility of the system design resulting from the use of graphical models for representing system structure.

The structured approach is irrelevant and obsolete given modern development trends (Yourdon, 1988, Yourdon, 1991). It is a process-oriented approach and process-oriented approaches suffer from a single fundamental problem; they are volatile and prone to change. Data driven approaches in contrast, are less volatile and represent a more stable underpinning for systems development and consequently, a number of methodologies based on this new approach emerged.

- **The Data Driven Approach**

Warnier (1976) and Orr (1977) were among the first researchers to propose a data driven approach to systems development. Orr's data-structured systems development (DSSD) methodology was based on the realisation gained from practical experience that successful systems are often ones in which the program structure mirrors that of the data. What began as a program-design methodology focused on working backwards from data outputs to ideal outputs, subsequently evolved into a full systems development methodology dealing with database design, requirements definition, and systems planning and architecture (Orr, 1989).

One of the most popular data driven approaches was Information Engineering (IE). Its origins are not universally agreed upon, however, the work of Martin and Finkelstein (1981) and Martin (1982) is generally cited. The basic principles of IE are the following: data structures are quite stable and lie at the centre of modern data processing; communication among developers and users is facilitated by the use of graphical representations, and an overall organisational strategic orientation is necessary in systems development.

A fundamental precept of Information Engineering is that data is an important organisational resource and should be managed accordingly. In order to eliminate the data redundancy problems inherent in systems developed around functional processes, IE strongly recommends that organisations create a co-ordinated approach by having an overall enterprise model which can form the basis for all systems development (Finkelstein, 1989).

Although the data driven approach was a comprehensive guide to systems development, practitioners were slow to adopt it. The reasons for this stunted adoption were more commercial than functional and thus cannot be attributed to the usefulness of the data driven approach.

- **Object Oriented Approach**

In comparison with the more traditional systems development approaches which tended to adopt either a process driven or data driven perspective, the object oriented (OO) approach requires an entire paradigm shift. This is because it does not make a distinction between data and process (Coad & Yourdon, 1991). Instead, the attributes of the data structure and its methods are encapsulated into a single entity called an object. The principle of encapsulating data and methods into an object is similar to the concept of information hiding (Parnas, 1972). The internal structure of the object is thus hidden from other objects and can only be accessed by the methods of the object itself. This provides a degree of advantage in terms of facilitating reusability and maintenance.

The encapsulation of data and methods in an object causes a need for communication between these objects. This is brought about by the use of messages which are passed between objects requesting a particular method be performed. Coupling is low since the message does not need to know how the actual operation will be performed. This is known as polymorphism because a generic message can be sent to several different objects without the need for specifying implementation details. Polymorphism allows change to be accommodated more transparently.

There are a number of advantages related to the use of the object oriented approach (Fitzgerald *et al.*, 2002; Mathiassen *et al.*, 2000):

- The OO approach does not make the artificial distinction between data and process and because of this, a more natural modelling of real world problem domains is possible.
- Improved communication is experienced between developers and users since the users can understand a model based on the real world business context, better than a purely technical one. This aids in avoiding costly requirements specification errors.
- The need for separate representation models for analysis and design is eliminated in the OO approach. A single representation is used throughout the life cycle, with terms and concepts defined during analysis being standardised and used right up to the eventual program coding.

- A library of fully-tested objects which can be modified contributes significantly to reuse, which in turn improves productivity. The reuse of objects also reduces the need for maintenance in the future.
- The OO approach achieves a low level of coupling between objects and a high level of cohesion within objects through the mechanisms of encapsulation and polymorphism. As a result, changes are much more localised and easy to deal with and do not result in ripple effects being caused throughout the system.

The OO approach is not considered a new approach and has been regarded by many researchers as an evolutionary outgrowth of traditional approaches (Martin and Odell, 1992). The advantages of object orientation are certainly significant; however, its use in the commercial mainstream of business information processing has been limited. This limited use can be attributed to the fact that OO applications have been the domain of computer scientists and academics that have used it in areas of interest to them. These researchers are typically lacking the knowledge related to the business problem domain and have not applied OO to these areas. Researchers however argue that OO methodologies need to evolve further before they can cater for business situations (Thomann, 1994).

- **Prototyping Approach**

The primary problem with the SDMs described in the prior discussion is that user involvement is very limited. As a result, the user is not kept informed of progress made in the overall development of the system. Another major problem is that the methodologies based on the SDLC approach rely on the assumption that a complete and detailed understanding of the problem situation can be achieved in advance, and used throughout the development of the system, without providing leverage for any change.

The requirements definitions phase is arguably the most difficult part in systems development and the assumption that requirements can be determined completely in advance is “fundamentally wrong” (Brooks, 1987). Users are themselves unsure of the requirements of a system right from the beginning and there is inevitable backtracking as development progresses.

The evolutionary nature of systems requirements necessitates a high level of communication between developers and users. Users are often disenchanted once requirements are specified to the developers because they don't physically see any progress being made on the system and have to wait between eighteen months and three years for a finished product.

The prototyping approach attempts to counter some of these deficiencies by having an iterative process of demonstration, review, refinement and expansion. This provides the necessary leverage for change in requirements. In addition to this, users are able to "see" the progress being made in the development of the system since a prototype is an early version of the system that exhibits the essential features of the final product (Alavi, 1984).

The prototyping approach lends many benefits to the process of software development. These include (Fitzgerald *et al.*, 2002):

- Facilitating the specification of system requirements with greater precision and by so doing, allowing for validation before continuing with later development stages. Each successive prototype forms a closer approximation of the requirements.
- Prototyping helps to increase user participation and commitment, which leads to increased user acceptance of the system. Users also participate more freely and developers achieve better communication and establish greater rapport with the users.
- The quality of systems should show an improvement since requirements are more complete. Users are consequently more satisfied and less likely to request immediate modifications to systems when they are delivered.
- The prototyping approach allows developers to demonstrate a functional system quickly and this creates the illusion that the system is being developed rapidly.

The strengths mentioned above advocate the prototyping approach as a seemingly improved alternative to conventional approaches. However, there are a number of weaknesses that have been identified (Fitzgerald *et al.*, 2002):

- Projects based on the prototyping approach do not necessarily have clearly prescribed phases with milestones and specific deliverables. This makes them more difficult to manage than projects undertaken with conventional approaches.

- Prototyping can lead to a considerable waste of time and resources if the prototype is inappropriate or untimely. As a result of this, the development team may not make any real progress in developing the system.
- In the haste to develop a prototype, the formulation of an overall systems plan may be overlooked and this can lead to problems with systems integration.
- The speed at which a prototype is delivered can create unrealistic expectations as to the time scale for developing fully-functioning systems amongst users.
- Large projects pose a potential problem since it is not clear how the system should be divided for the purpose of prototyping.
- Prototyping requires significant commitment on the part of users and it may be difficult to maintain user enthusiasm beyond the initial specification of high-priority requirements.

Prototyping provides a fresh and new approach to systems development but essentially, it is also based on the SDLC methodology with more emphasis being placed on interaction with users and a longer lifecycle.

The methodology era was characterised by the development and use of systems development methodologies that had a structured, formalised and documented development process. Methodologies that stemmed from this era were all primarily based on principles of the systems development lifecycle with the development process being broken into a number of discrete phases, typically beginning with planning or requirements gathering and ending with the implementation of the operational system.

The constantly evolving arena of systems development has necessitated the re-assessment of previously used methodologies. Changing development environments together with a more competitive software industry have imposed shortened development times on systems development projects. Methodologies that were used in the methodology era are as a result, not ideally suited to such an environment. It is purported by Avison and Fitzgerald (2003) that we have moved into a post-methodology era which began in the late 1990's. They state that this era has been characterised by a serious reappraisal, by researchers and practitioners, of the concepts and usefulness of earlier methodologies.

2.6.4 **The Post-Methodology Era**

During the methodology era, systems development methodologies were promoted as being a cure-all for software development project woes. The use of a methodology was considered to be a vital contributor to the success of a development project.

However, after using and experiencing methodologies that emanated from the methodology era, many organisations are still looking for alternative methodologies to assist in the development of their software. Some organisations have even gone to the lengths of abandoning the use of methodologies completely (Avison and Fitzgerald, 2003).

This has prompted researchers to suggest that software development has entered into an era where methodology use has deviated from the traditional SDLC-based methodologies and perhaps even dwindled slightly.

There are a number of reasons for the lack of use and interest in methodologies (Avison and Fitzgerald, 2003; Vickers, 1999):

- Perhaps the most common reason would be disappointing productivity. Developers expect their productivity levels to increase drastically with the use of a methodology however, this is not always the case.
- Methodologies are overly complex since they were designed for large and intricate development projects. This complexity may lead to the definition of extraneous requirements that form part of user wish lists instead of legitimate requisites.
- The use of methodologies requires highly technical skills that may be costly and difficult for developers and users to acquire. Additionally, the tools related to the use of methodologies may be difficult and costly to use, yet still not deliver adequate benefits.
- Methodologies are often not dependant on the type or size of the project, nor upon the technological environment and organisational context. A single rigid methodology adopts only one approach to the development of projects and may not address a particular organisation's underlying issues and problems. It is thus unacceptable to assume that a methodology is universally applicable.
- A methodology may be inflexible and provide no room for changes to be made to requirements during development.

- Some organisations, after using a methodology, still do not advocate it, citing that it either led to a poorly developed system, or that it actively hindered the system from being developed.
- The risk involved in implementing a methodology which ultimately may not work for an organisation, also contributes to the aversion to methodologies.
- Methodologies are often very technical and do not take the human aspects of a software development project into consideration. These aspects are often as important as the technical aspects are to the success of a project.

Methodologies have been evolving since their inception into the field of systems development. However, the post-methodology-era is more than just an evolution of the methodology. It has questioned the actual use and effectiveness of methodologies and thus merits more attention and research.

New ways of presenting software, such as web-based applications, have emerged during this era and they require new methodologies to provide guidance in developing them. The following sections will describe some of the newer methodologies that have emerged during the post-methodology era and discuss the impact that they have had on software development.

2.7 THE DAWN OF A NEW ERA: CURRENT AND FUTURE TRENDS OF SYSTEMS DEVELOPMENT METHODOLOGIES

The post-methodology era has seen the emergence of a number of new methodologies such as component-based development, extreme programming, and web development methodologies such as WISDM and web engineering. These methodologies will be discussed next.

2.7.1 Component-Based Software Development

The time and resource constraints on the development of software have become more rigid in the post-methodology era. Component-based development is being regarded as an important addition to the set of existing development methodologies to assist in meeting these stringent constraints adequately, and in a timeous manner (Vitharana *et al*, 2003).

Grounded in the concept of component fabrication and assembly, component-based software development (CBSD) can help the software industry realise quality and productivity gains similar to those achieved in the hardware and manufacturing industries (Brown, 2000; Szyperski, 1998). With the number of software projects that miss schedules, exceed budgets and deliver defective products rising, industry experts are turning to component-based solutions to overcome the current software crisis (Fayad and Schmidt, 1997).

The CBSD methodology seeks to develop the components required to support various functions and processes for a particular domain or area (Vitharana *et al*, 2003). A component is defined as a piece of executable software that offers a set of related services through a published interface (Hopkins, 2000; Vitharana *et al*, 2003).

There are various domains which can be selected for componentising. Once the components for that particular domain are designed and developed, various applications are constructed by first selecting and then assembling chosen components.

A domain might contain hundreds or even thousands of components and for this reason, some method of systematically cataloguing the components for search and retrieval is essential. The classification and coding of components in a knowledge base or repository assists application assemblers to easily find the components that match user requirements and support the desired function or process of a particular domain application.

The lifecycle of the CBSD methodology can be visualised as follows:

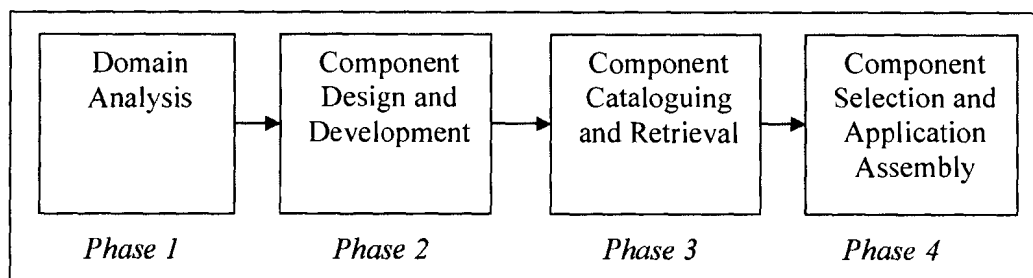


Figure 2.1. CBSD Lifecycle (Vitharana *et al*, 2003)

- **Domain Analysis** – During this phase the application domain is analysed and objects representing the domain are identified. These objects are then used as building blocks for the construction of components.

- ***Component Design and Development*** – Component design can be likened to manufacturing since variables such as cost effectiveness, ease of assembly, reusability, customisability and maintainability, are considered when designing components.
- ***Component Cataloguing and Retrieval*** – Designing the correct components related to a particular domain in CBSD is important, however, it is also vital that component assemblers are able to discover the appropriate components when constructing customer applications. Since the need for an application is driven by a set of user requirements, finding the components that match those requirements is critical to delivering the desired product. As a consequence, a component repository or knowledge base is set up during this phase.
- ***Component Selection and Application Assembly*** – With the availability of a knowledge base, users and analysts uncover relevant application requirements as they navigate through it and at the same time increase their knowledge about the problem domain (Mili *et al*, 1997). Since the knowledge base holds the domain knowledge, it will eventually contain the best practices of that domain or industry (Vitharana *et al*, 2000). Users are also able to actively participate in requirements definition since they themselves identify the software components that satisfy their needs.

It is evident that component-based software development can offer many benefits in software development efforts. The following key advantages have been identified (Vitharana, 2003; Clements, 1995; Haines *et al*, 1997):

- ***Reduced Lead Time*** – Building complete business components from an existing pool of components significantly reduces development time.
- ***Leveraged Costs Developing Individual Components*** – The cost of developing applications is also reduced since components are reused in multiple applications.
- ***Enhanced Quality*** – Reusing components in multiple applications also ensures that they get tested in various implementations and circumstances.
- ***Maintenance of Component-Based Applications*** – Components that have become obsolete can be easily replaced with new and enhanced ones.

There are however, also many challenges related to the use of CBSD (Crnkovic and Larson, 2002):

- ***Component Generality and Efficiency*** – The principle of reuse is beneficial to development and maintenance of applications, but it places a high demand on reusable components. The components are required to be sufficiently general to cover an entire domain and different aspects of their use, but they should also be simple and concrete enough to serve a specific requirement in an efficient way.
- ***Evolution of System Requirements*** – Components need to be easily adaptable to continually changing requirements which have become characteristic of the post-methodology era.
- ***Compatibility*** – An important factor for the successful reuse of components is the compatibility between different versions of components. Ideally, a component can only be replaced easily, or added to new parts of a system, if it is compatible with its previous version. The development of software is extremely complex and changing the functionality of a component in the slightest manner may dramatically impact its compatibility with the rest of the system.
- ***Testing*** – Testing a component that will eventually form part of an entire system is difficult, since the exact nature of the end product is often not known during the development of the components. Rigorous testing needs to be done so that components that are developed don't become obsolete due to poor design and planning.
- ***Maintenance*** – The process of maintenance is now more complex since it needs to be handled at three different levels: The system level where problems are reported, the product level where errors in a specific product version are reported and finally on the component level where the fault is located.

CBSD is a comparatively new methodology and warrants more research. It does seem to address some of the issues that are present in the post-methodology era by reducing costs, development time and improving the quality of systems. There is anecdotal evidence that suggests that the CBSD methodology does indeed work and yield successful results.

Notwithstanding this, the CBSD methodology deviates completely from traditional lifecycle methodologies that were developed and used in the pre-methodology and methodology eras. As hard as it may be for software developers to make the paradigm shift that is required when using component-based software development, it will become a necessity when the reality of the post-methodology era dawns on them.

2.7.2 Web Application Development

The post-methodology era has been characterised by a need for rapid application development due to the competitive environments in which organisations find themselves. The Internet has changed software development's priority from *what* to *when*, and reducing the development cycle is now one of software engineering's most important missions (Aoyama, 1998).

E-commerce – the leading Web-based application – is projected to have a market exceeding one trillion dollars over the next several years (Joshi *et al*, 2001). This is indicative of the number of web-based applications that have been developed recently. Although these applications are being deployed in large numbers, they are being developed in an arbitrary, anarchic and ad-hoc manner. Effective ways and techniques for ensuring the integrity and maintainability of web applications are thus lacking (Murugesan, 1999). Although there are a few web development methodologies such as WISDM, Internet Commerce Development Methodology (ICDM), e-RAD, web engineering and Web XStream, they are not used adequately enough.

A component-based development methodology that would effectively manage the development of Web applications has been suggested by Lee and Shirani (2004). A component, in the context of Web-based applications, is a system or a software program that has been pre-compiled to provide certain functionality. To facilitate the use of a component-based methodology, a web application can be considered as an assembly of compendia with each compendium consisting of one or more pages. Each page can be visible or invisible and contain interactive elements.

Based on these divisions, the web-based application can then be componentised and the phases described in section 2.6.1 can be followed to develop it in a manner that exclusively utilises components.

- **Web IS Development Methodology**

Web IS Development Methodology (WISDM) is another methodology that has been designed specifically for the development of web-based applications. Figure 2.2 shows a conceptual model of this methodology.

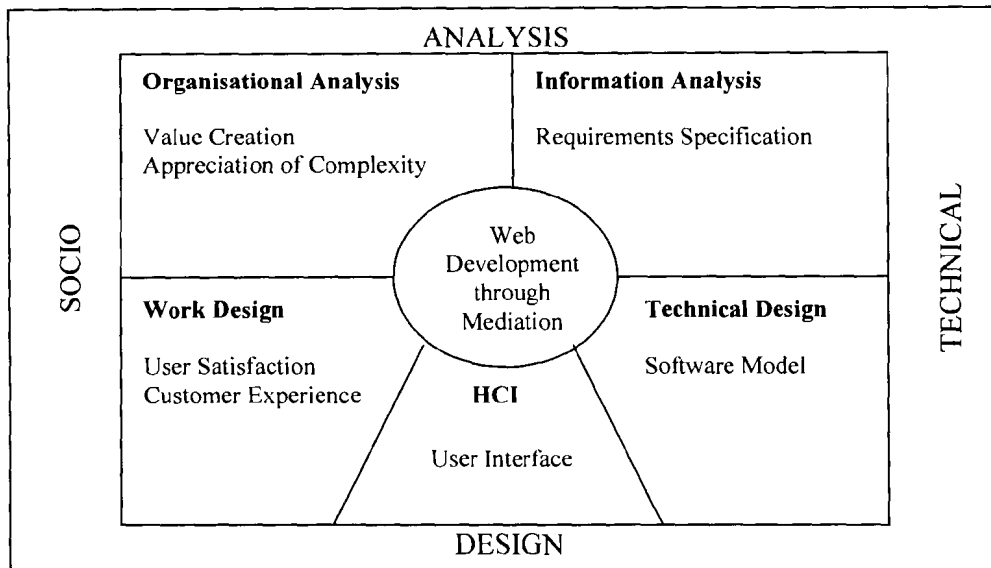


Figure 2.2 WISDM (Avison and Fitzgerald, 2002)

The basic concepts of WISDM are discussed briefly below (Avison and Fitzgerald, 2002):

- *Organisational Analysis* – This represents value creation and relationship building with a broad range of stakeholders including customers, employees, government, suppliers, labour organisations, etc. A wide variety of people use the website and it is thus essential to maintain a relationship with as many of the stakeholders as possible.
- *Information Analysis* – Requirements specification is represented by this facet of WISDM. This entails the formal specification of the information and process requirements of the organisation in the form of a document, graphical representation or software prototype.
- *Work Design* – This represents user satisfaction and attempts to incorporate the interests of external users who may use the information system as part of their social activities.
- *Technical Design* – A software model which details the data structures and program design that is required for the construction of the application.

- *Human-Computer Interface (HCI)* – The user interface is the focal point of this facet of WISDM. It is represented by an overlapping space between technical design and work design and its conical shape signifies its role in pointing towards analysis, but with its foundations in design. Page layout, navigation schemes and security are emphasised in the process of designing the user interface.

With web-based applications becoming increasingly popular, it has become necessary for some sort of guiding methodology to aid in the development of such applications. The primary reason for developers using ad-hoc methods for development of web applications is that they are uneducated in the creation of applications of this nature (Retalis *et al*, 2000). It is thus imperative that training be provided on web application development to ensure that this widespread implementation of software continues to proliferate.

2.7.3 **Extreme Programming**

Created by Kent Beck in 1996, the purpose of Extreme Programming (XP) was to fulfil a need for a faster, simpler and cheaper way to design software. Extreme Programming can also be considered as being part of the post-methodology era since it is radically different to the long-established methodologies. The following twelve practices make XP uniquely different to other methodologies (Lippert and Roock, 2001; Munro, 2003; Nuckols & Canna, 2003; English, 2002; Newkirk & Martin, 2000):

- ***Pair Programming*** - Two developers program at one workstation together. Programmers switch seats periodically and regularly discuss each others code. This increases the number of people that are familiar with the source code.
- ***Continuous Integration*** - Small changes in the code are frequently integrated into the common source base on a daily basis. Integration of changes, one set at a time, simplifies the integration process and makes it obvious who is responsible for fixing the code when integration tests fail.
- ***Collective Ownership*** - The code and all development documents are owned by the entire team. Any member is free to modify any part of the documentation or code at any time. This approach is very different to the traditional one where a single developer owns a set

of code. XP supporters argue that the more people that work on a piece, the fewer bugs that will occur.

- ***Small Releases*** - The system is planned with short release cycles containing the most valuable business requirements. Typically, one cycle is less than three months and this allows the user to see and touch the working product on a regular basis.
- ***Metaphor*** - The system is based on a small set of powerful metaphors which guide the developers through the development process and explain the systems behaviour. They typically take the form of customer stories which are high-level requirements of a system written in simple prose instead of the more traditional, and technical Unified Modelling diagrams and models. Metaphors express the evolving project vision that defines the systems scope and purpose.
- ***Planning Game*** – This stage encompasses the requirements definition phase of traditional SDMs but it is considerably different to the traditional method. The user lists the requirements of the system with “stories” written on index cards. These stories basically tell the developers what the user expects from the working system. The developers then provide estimates on how long each story will take to complete and the approximate cost of each. Based on the estimates made, the user then chooses which story will be implemented completely in an iteration and agrees not to make any changes or additions to it until the iteration completes. An iteration is 1-3 weeks long and a small release is made up of a number of such iterations. The planning game occurs after each iteration and until the system is fully implemented.
- ***Testing*** - Two types of tests are carried out continuously. First, unit tests ensure that classes do what developers expect them to do. These tests are typically written by the developer. Second, acceptance tests, written before the code that they will test, ensure that the system does what the user expects it to do. These tests are derived from the customer “stories” or scenarios. All code has an associated test and new tests are added to old ones in a testing framework, creating a comprehensive test suite.

- **Refactoring** – If a design is no longer appropriate, it is changed. This involves the restructuring of the software without changing its behaviour (Munro, 2003; English, 2002). The combination of the test suite with refactoring lets the programmers avoid the classic “fix one thing, break another” bug.
- **Coding Conventions** – A uniform style of code is used and this simplifies collective code ownership. All classes and methods are named in a standard way which is easily understood by the entire team, and a standard architecture is used to facilitate reuse of code. The programmers follow these rules so that all code looks as if it were written by a single person.
- **Simple Design** - Simple designs are used because they are easy to create, maintain, understand and they discourage complexity. The traditional method of building “hooks” for future functionality often leads to large systems with substantial elements that are initially unused and often discarded. In XP, if the design falls short of the required purpose, it is refactored.
- **On-site Customer** - This ensures that the developers are always concentrating on the requirements and if they lose focus of these requirements, the customer is at hand to put them back on track.
- **40 Hour Week** – Overworking developers is not encouraged. Well rested developers are motivated and boost productivity.

The twelve principles discussed need not be followed to the letter; instead they act as guidelines to developers during the development of systems. As a consequence, these core practices can be adapted and modified by developers to accommodate systems development in their organisation.

With the post-methodology era being characterised by a requirement for adapting methodologies, XP may indeed be a methodology that addresses this need.

2.8 DISCUSSION AND FINAL COMMENTS

The post-methodology era has brought with it new ideas and concepts, such as Extreme Programming and CBSD that many individuals will find difficult to grasp. These changes are all part of the continuous evolution of the software industry and software methodologies in particular. For some developers the post-methodology era represents the abandonment of methodologies altogether; for others, it represents improved methodologies while moving away from the highly bureaucratic types of methodologies of the methodology era; while for others it represents moving beyond in-house development altogether.

The post-methodology era is characterised by diversity but this should not be interpreted as the death of the methodology. Many traditional methodologies are still being used along with alternative methodologies. Importantly, even with developer backlash against formal methodologies, they remain influential in practice. They might be adapted or other tools and techniques may be used, but they still contribute in practice. In light of the diversity of this era, organisations are much more likely to find an appropriate methodology for their systems development work even though they rarely find a single clear strategy for doing so.

The risk for organisations not using any methodology at all should be recognised and the lessons of history not completely ignored.

CHAPTER 3

CONCEPTUAL RESEARCH MODEL AND RESEARCH METHOD

3.1 INTRODUCTION

This chapter will, at the outset, discuss the theoretical background of the study. Since this study is an extension on the work done by Huisman (1999), the conceptual research model is one that was utilized by Huisman (1999) and is based primarily on the classical diffusion of innovations theory of Rogers (1995). This will be discussed first.

Following this will be a discussion of the research method used and the response received to the survey.

3.2 CONCEPTUAL RESEARCH MODEL

The conceptual research model presented in Figure 3.1 is analogous to the one used by Huisman (1999). This model was deduced from existing literature as follows. The classical diffusion of innovation theory of Rogers (1995) was used as the main theoretical framework. This was then combined with some theories on innovation diffusion in Information Systems, and other complementary theories. The conceptual research model consists of three components, namely the innovation, the innovation diffusion process and the adopting unit. Each component will be discussed next.

3.2.1 The Innovation: Systems Development Methodologies

Systems development methodologies are predisposed to focus on the technical task of the IS department (Swanson, 1994). There is also a high level of complexity related to systems development methodologies since developers need to overcome a high knowledge burden before being able to use them effectively (Fichman, 1992). The development of systems is normally performed by teams of individuals which imply high user interdependencies.

This study will perform an investigation into the use and effectiveness of systems development methodologies in organisations and amongst individual developers.

The characteristics of systems development methodologies which will be studied are perceived relative advantage, compatibility, complexity, trialability and observability (Rogers, 1995).

3.2.2 Innovation Diffusion Process

The main focus of this study is the use and effectiveness of systems development methodologies, which is represented by the implementation and confirmation stages of the innovation-decision process as described by Rogers (1995).

The description of McChesney and Glass (1993) is used in the conceptual model since it adequately describes the post-implementation stage and incorporates the use and acceptance of the systems development methodology.

3.2.3 Adopting Unit

The social system under examination is an organisation. Focus will be drawn to all the elements of the social system as discussed by Rogers (1995). These include the structure of the social system, its norms, the opinion leaders and types of innovation decisions, and the consequences of innovations.

The relationship between the deployment of SDMs and the size, maturity and culture of the IS department will be studied as part of the structure and norms of the social system. The consequences of innovations will also be studied by conducting an investigation into the perceived support provided by a systems development methodology and its impact on both the developed system, and the development process.

The difference between the adoption and acquisition of a technology at the organisational level, and its adoption and implementation at the individual level, needs to be taken note of (Dietrich *et al.*, 1997).

This difference has an important consequence on the identification of the adopting unit in the social system, since it implies the existence of multiple adopting units. For the purposes of this study, two adopting units, namely the IS department and the individual systems developer have been identified.

To facilitate the description of the adopting units, the classification proposed by Kwon and Zmud (1987) is used. This proposed classification consists of individual, organisational (structural), task and environmental characteristics. These characteristics are indicated in the conceptual research model in Figure 3.1.

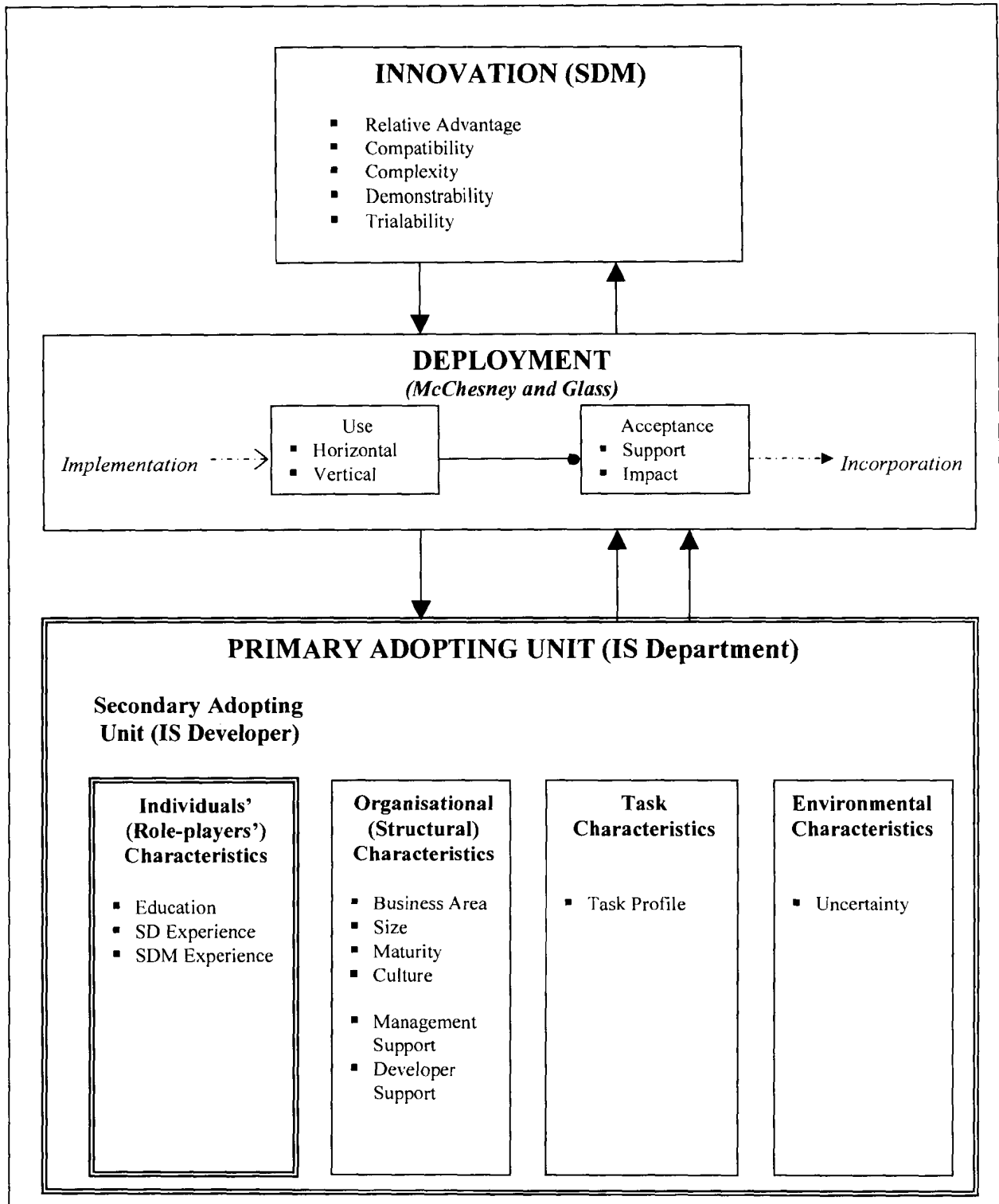


Figure 3.1 Conceptual Model of the Dissertation (Huisman, 1999)

Despite the limitations of the classical innovation diffusion theory, it still remains a prevalent reference theory for empirical studies of information technologies. It has aroused a lot of interest in the IS community and researchers (Kwon and Zmud, 1987; Fichman, 1992; Iivari, 1993; Swanson, 1994; Prescott and Conger, 1995; Huisman, 1999; Beynon-Davies and Williams, 2003) have reviewed existing studies into the innovation diffusion theories in IS. These papers indicate that researchers have challenged and extended the work of Rogers and simultaneously integrated it with other theories.

The attention that it has received is indicative of the fact that the innovation diffusion theory is an accepted reference theory in empirical research and that it will continue to be used for such studies in the future.

3.3 RESEARCH METHOD

3.3.1 Development of the Questionnaire

The domain of system development methodologies is constantly evolving and continuous research is necessary to gauge whether the changes being made are indeed beneficial to the discipline. Consequently, this study extrapolates on the work done by Huisman (1999) and seeks to identify changes that have occurred in the development of systems since then.

To maintain continuity between this study and that of Huisman's (1999), the format of the questionnaire was the same as that of Huisman's i.e. a multi-respondent survey. Since the questionnaire retained its format and was proven to be a reliable instrument of measurement during its development in Huisman's study, it was deemed unnecessary to re-measure its reliability. Huisman's study used a paper-based, mail format questionnaire, whereas this study utilised an electronic questionnaire that was distributed via e-mail. The questions contained within the survey were similar to those in Huisman's; however they were modified to include certain new technologies that had come to the fore since 1999.

The design of the questionnaire was carried out in collaboration with Statistical Consultation Services at the North West University's Potchefstroom campus. Consultation with abovementioned department was required to determine the best format for the questionnaire, since its format was changed from paper-based, to electronic.

An electronic format for the questionnaire was decided upon due to time constraints and the problems with the mail delivery system in South Africa. The five-point Likert scale used by Huisman was retained in this questionnaire since it is the most frequently used response scale (Conger, 1994). Discussions with Statistical Consultation Services aided in gaining some insight into the advantages and disadvantages of using an electronic questionnaire.

These are summarised briefly below:

Advantages

- Since there is no requirement for paper, envelopes or stamps for sending out electronic questionnaires, the cost-benefits in comparison with a regular mail survey are considerable (Bachmann & Elfrink, 1996; Couper, 2000; Llieva *et al.*, 2002; Yun & Trumbo, 2000).
- Unlike a mail survey, the time between questionnaire send-out and delivery is no more than a few seconds or a few minutes at the most. This reduces the total time it takes to conduct a survey and equates to a monetary saving as well (Bachmann & Elfrink, 1996; Garton *et al.*, 2003; Taylor, 2000; Yun & Trumbo, 2000).
- The ability to send the exact same message at exactly the same time to multiple recipients is a further advantage.

Disadvantages

- Electronic questionnaires may pose some difficulty to individuals that don't have much experience using a computer.
- E-mail and conventional mail are treated very differently by people. Lengthy drawn-out e-mails are usually discarded by readers after reading the first few lines if the e-mail was found to be disinteresting. It is also possible that the message can be trashed without even opening it (VanDenKerkhof *et al.*, 2004)
- The response rate of a questionnaire sent via e-mail has been shown to be markedly lower than that of a questionnaire distributed via the postal system (Mehta and Sivadas, 1995; Stanton, 1998).

The aspects above were taken into consideration when designing and distributing the questionnaire in order to attain a set of results which would prove to be beneficial to this study.

The questionnaire was designed with functionality, compatibility and usability in mind. It was subsequently implemented as a Microsoft Excel spreadsheet containing macros. A certain level of *functionality* was attained by capturing the responses provided by the participants in a separate sheet in the Microsoft Excel workbook. The controls on the questionnaire consisted of text-boxes, radio buttons and check-boxes. With the aid of macros, each time a particular control was selected, the appropriate number of the question with its related answer was written to a second worksheet in the same Excel workbook. To prevent the user from tampering with the answer sheet, it was password-protected and hidden from view.

The questionnaire was presented in a Microsoft Excel format since a vast majority of organisations use the Microsoft Office suite of products. By doing so, *compatibility* issues were eliminated to a certain degree and participants were not impeded in completing the questionnaire. This, together with the single-click method of answering the questions also assured *usability*.

3.3.2 Distribution of the Questionnaire

The target population for this questionnaire were organisations that developed software in-house. The ICT Users Handbook of 2003 was used as it was the most comprehensive source for such a listing and more importantly, because it was also used in Huisman's study in 1999. As a result, continuity between the two studies could be sustained.

Before emailing the questionnaires to the various companies contained within the ICT Users Handbook, they were contacted telephonically to briefly describe the nature and purpose of the survey. During this telephonic conversation, it was also determined whether the company in question did indeed develop software and if so, whether the questionnaire would be applicable to the type of development that they were doing. The contact person at each of the companies was the IT manager.

A total of one hundred and thirty six companies were contacted telephonically to determine whether they would be willing to participate in the survey. of the one hundred and thirty six, sixty one agreed to participate in the study. The questionnaires, together with a cover letter explaining the purpose of the questionnaire in more detail and instructions on how to complete it, were e-mailed to each IT manager that agreed to contribute to the study.

Organisations that did not return the questionnaire within two weeks were sent reminders. A maximum of three reminders were sent to non-responsive organisations before they were excluded from the study. Figure 3.2 gives a summary of the number of companies contacted and the response received.

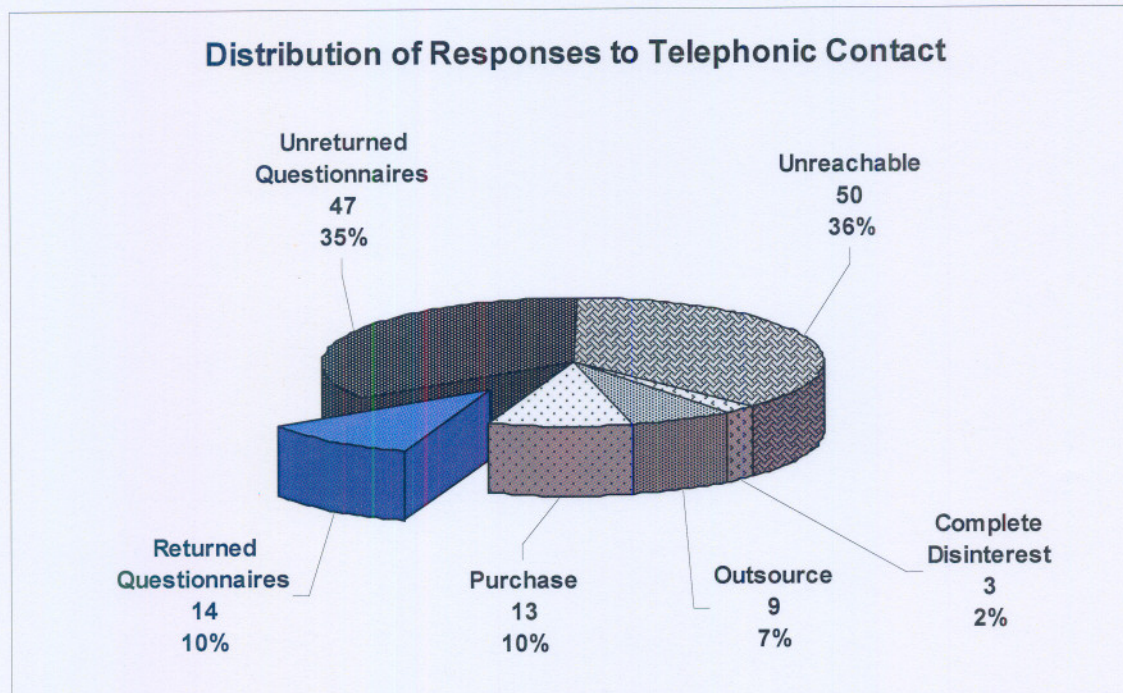


Figure 3.2 Distribution of Responses to Telephonic Contact

Figure 3.2 shows the distribution of the distinct categories of responses received to the survey. The responses illustrated in the figure encompass those received during telephonic and e-mail contact with the IT managers at each of the one hundred and thirty six companies. Each category of response will now be briefly elaborated upon.

- **Unreachable**

Some organisations' IT managers could not be reached via telephone because either their number had changed or they were unavailable. In cases where they were unavailable, a second phone call was placed at a later stage however, they still remained beyond reach. In total, 36% of the organisations could not be reached.

- **Complete Disinterest**

Companies falling into this category refused to participate in the study and did not cite any reason for doing so. 2% of the organisations refused to participate in the study. No further correspondence was carried out with them.

- **Outsource and Purchase**

Companies that do not develop their own software and outsource all of it, or purchase commercially available off-the-shelf software, fall into this category. These companies were only contacted telephonically and it was at this first contact that it was determined they did no development of software or use a methodology. Due to this fact, 7% of the organisations were not sent a questionnaire.

- **Returned questionnaires**

Organisations that returned questionnaires form this category. of the total population contacted, 10% of the organisations returned the questionnaires that were sent to them.

- **Unreturned Questionnaires**

This category consists of IT managers that agreed to participate in the survey and received the questionnaire, but did not return it. 35% of the organisations fall into this category. A further break-down of this category is shown in figure 3.3.

The discussion above was based on the entire population, i.e. the 136 organisations contained in the ICT Users Handbook of 2003 that were contacted telephonically. Figure 3.3 shows a categorisation of the 61 companies that agreed to participate in the survey and received the questionnaire via email.

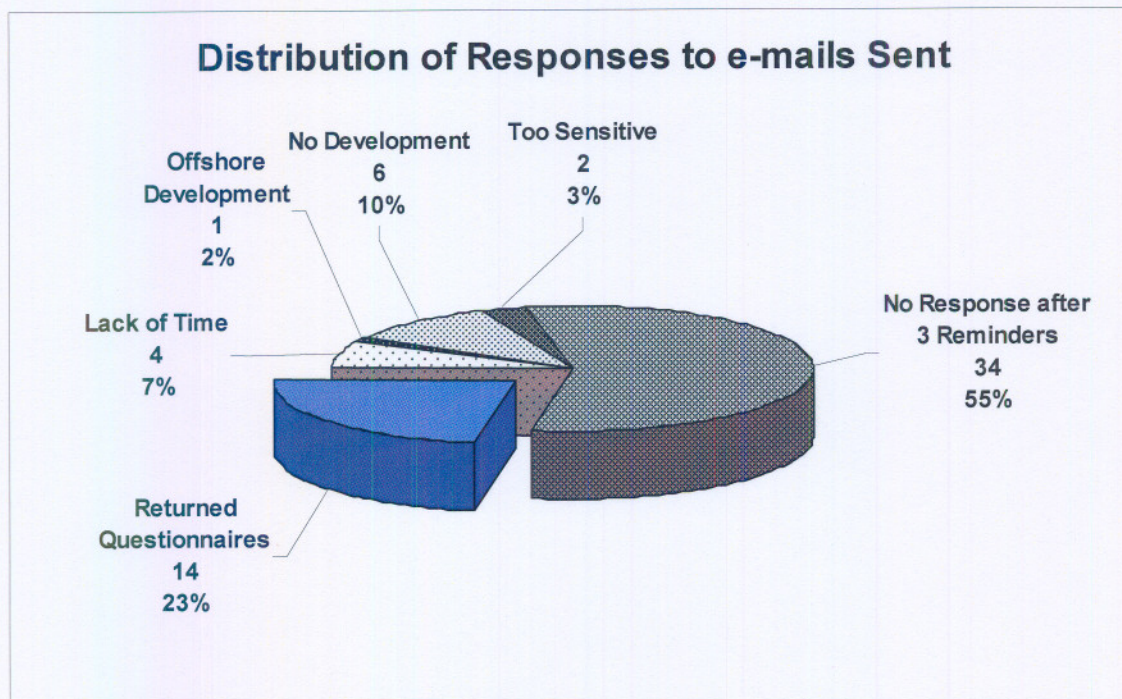


Figure 3.3 Distribution of Responses to Questionnaires e-mailed

Each category of response will be elaborated on next.

- **Returned Questionnaires**

A total of fourteen out of the sixty-one organisations (23%) that were sent questionnaires returned them. The questionnaires were all received within a period of 3-5 weeks after being sent.

- **Lack of Time**

Four organisations (7%) agreed to complete the questionnaire during the telephonic contact but subsequently cited a lack of time for not completing it. These organisations were all sent numerous reminders via e-mail before giving this reason for not returning the questionnaire.

- **Offshore Development**

One organisation (2%), after receiving the questionnaire and perusing through it, revealed that most of its software development occurred outside of South Africa. Since this is a study whose focal point is South Africa, the questionnaire would lose its relevance if it were sent offshore for completion. No questionnaire was thus received from this organisation.

- **No Development**

Six of the organisations (10%) that received the questionnaire stated that after reviewing the questions contained within it, it became evident that the questionnaire was not relevant to their business model i.e. they did not do any major software development and their IT departments consisted of a maximum of 3 to 4 people. These organisations therefore refrained from completing the questionnaires sent to them.

- **Sensitive Questions**

Some organisations (3%) reluctantly agreed to participate in the survey since they did not wish to reveal any sensitive business information while completing the questionnaire. They did nevertheless agree to receive the questionnaire however, after examining it, decided that they would not be comfortable completing it due to the nature of the questions contained in it and for fear of revealing trade secrets.

It must be re-iterated that all the organisations were assured that their identities would remain anonymous, and all information obtained from the questionnaire would be utilised for research purposes only.

- **No Response after Three Reminders**

Organisations that consented to receiving the questionnaire but were never heard from again are contained within this category. Despite being sent three reminders over a period of two months to urge them to complete the questionnaire, no response whatsoever was received from them. 55% of the organisations did not respond after being sent three reminders.

3.4 DISCUSSION AND FINAL COMMENTS

This chapter presented the conceptual model of the survey and the response rate received to the questionnaire. The conceptual model is identical to the one used by Huisman (1999) since this study investigates the changes that have occurred in the use of systems development methodologies since then, and it uses the same measures to determine the degree of change.

The response rate of twenty-three percent was low, however it did allow for a satisfactory investigation into the use and effectiveness of systems development methodologies in South Africa.

A contributing factor to the low response rate was that several organisations outsourced their systems development or purchased off-the-shelf software. Additionally, many organisations used ERP systems and thus did not perform any software development. These two categories of response totalled approximately 20% of the entire population of one hundred and thirty six.

It was still possible to compare this study to the one conducted by Huisman (1999) even though this sample population was smaller Huisman's. This was made possible by using the appropriate statistical measures, as advised by the Statistical Consultation Services at the North-West University, for all analyses on the data.

CHAPTER 4

RESULTS OF DESCRIPTIVE STATISTICAL ANALYSES

4.1 INTRODUCTION

This chapter introduces the results of the statistical analyses that were carried out on the data gathered from the target population. The descriptive statistics that were calculated will be presented in this chapter. As the findings are explained, a comparison will be drawn with the findings that were obtained in the study performed by Huisman (1999). The instrument of measurement used was already proven to be reliable in Huisman's study and thus the composition of each measure constituting the questionnaire will not be explained. The results obtained from the instrument of measurement will be reported on.

Although the sample population for this study was somewhat smaller than Huisman's, the appropriate statistical measures were used to facilitate a valid basis for the comparison of results between the two studies.

4.2 DESCRIPTIVE STATISTICS

This section will provide details on the descriptive statistics that were performed on the data.

4.2.1 Profile of Organisations and Developers

Table 4.1 overleaf gives a summary of the profile of the organisations that responded to the survey.

Over a third of the organisations that participated in the survey stated "Manufacturing" as their business area with "Software Houses" coming second with just over 27%. Only 9% of the organisations were in the "Administrative Services" category. The profile of the responding organisations in this study is very similar to that of Huisman's. The only difference is that more software houses responded to this survey than to the one conducted in 1999. This similarity can be attributed to the fact that the same source for target organisations was used in both studies i.e. The ICT Users Handbook.

Type	Kalanjee		Huisman (1999)	
	Number (n=11)	Percentage (%)	Number (n=72)	Percentage (%)
<i>Business Area</i>				
Administrative Services	1	9.1	3	4.2
Software Houses/Software Consulting	3	27.3	5	6.9
Manufacturing	4	36.3	24	33.3
Other	3	27.3	23	31.9
<i>Organisation Size</i>				
1-50 Employees	2	18.2	5	6.9
51-200 Employees	1	9.1	8	11.1
More than 200 Employees	8	72.7	60	83.3
<i>IS Department Size</i>				0.0
1-5 Employees	5	45.5	17	23.6
6-20 Employees	2	18.2	23	31.9
20-50 Employees	1	9.1	12	16.7
More than 50 Employees	3	27.2	21	29.2

Table 4.1 Profile of Responding Organisations (n=11)

The majority of organisations that contributed to the survey were large organisations having more than two hundred employees. This was the case in Huisman's study as well.

Interestingly, the size of the IS departments is not proportionate to the organisation size. 45% of the IS departments employed one to five people and only 27% had more than fifty people employed. Compared to the findings in 1999, it seems as though the size of IS departments in the range of six to twenty employees has diminished somewhat, whereas IS departments with a maximum of five employees has increased. This could be an indication that IS departments require less staff now due to the automation of certain tasks.

A summary of the developer profile is shown in Table 4.2. A large number of the developers state their primary role as an "Analyst/Programmer". Only 13% of developers have a primary role of "Programmer". The distribution of roles has not changed much since Huisman's study.

The level of education amongst developers is centred mainly on a high school education and further education in the form of a Diploma. Only 27% of the developers have a university or technicon degree compared to over 30% in 1999. The table shows that although the developers have had an education after high school, not many developers pursued any post-graduate studies.

Type	Kalanjee		Huisman (1999)	
	Number (n=15)	Percentage (%)	Number (n=234)	Percentage (%)
<i>Title</i>				
Project Manager	1	6.7	25	10.7
Systems Analyst	1	6.7	24	10.3
Analyst/Programmer	11	73.3	93	39.7
Programmer	2	13.3	36	15.4
<i>Education</i>				
Senior Certificate (High School)	4	26.7	39	16.7
Certificate or Diploma	4	26.7	79	33.8
University or Technicon Degree	3	20	75	32.1
Honours or Masters Degree	2	13.3	36	15.4
PhD Degree	0	0	0	0.0
Other	2	13.3	3	1.3
<i>Experience in Systems Development</i>				
1 - 2 years	1	6.7	21	9.0
3 - 5 years	3	20	51	21.8
6 - 10 years	3	20	53	22.6
More than 10 years	8	53.3	89	38.0

Table 4.2 Profile of Responding IS Developers (n=15)

Although the developers' level of education is lower than was found in 1999, their experience in systems development is high, with 53% percent having developed systems for more than ten years. In total, over 73% of the developers had more than five years experience in systems development.

4.2.2 Characteristics of the Development Environment in IS Organisations

Having established the profiles of the organisations and its developers, the following paragraphs will discuss the development environment together with the development activities, and also present a description of systems development methodology use in South Africa.

Managers were asked to describe the development environment in their IS department. Table 4.3 shows the development environment for the IS departments as was indicated by the department managers in the questionnaire.

Type	Kalanjee		Huisman (1999)	
	Number (n=11)	Percentage (%)	Number (n=72)	Percentage (%)
<i>Development Platform</i>				
Microcomputers	7	63.6	50	69.4
Workstations	3	27.3	42	58.3
Mainframes	2	18.2	38	52.8
Minicomputers	2	18.2	33	45.8
<i>Networks</i>				
Local Area Network	9	81.8	69	95.8
Wide Area Network	3	27.3	51	70.8
Missing	1	9.1	3	4.2
<i>Operating Systems</i>				
Windows 3.x	1	9.1	12	16.7
Windows NT	1	9.1	55	76.4
Windows 95	1	9.1	57	79.2
Windows 98	5	45.5	32	44.4
UNIX	3	27.3	38	52.8
Windows XP	2	18.2	0	0.0
ICL VME	1	9.1	0	0.0

Table 4.3 Development Environment in the IS Departments

The most popular development platform amongst IS departments is micro-computers with a use of close to 64%. Workstations follow thereafter with a usage statistic of 27%. The use of mainframes has decreased drastically since Huisman's study, with approximately only 18% of IS departments using them presently, compared to almost 53% percent in 1999. This shows that development platforms have become more distributed in the last five to six years and that software development and maintenance need not only occur at a single, central location.

The majority of the IS departments use a local area network with the use of wide area networks only at 27%. The type of operating systems in use is worthy of note since 45% of IS departments still use Windows 98, 27% use UNIX and only 18% use Windows XP. This finding might be indicative of the fact that IS departments are comfortable using their current software and are reluctant to upgrade to a better version.

This could be due to the costs involved, or out of sheer aversion to change. Although these operating systems seem old relative to the newer ones available, operating systems such as Windows 3.x, Windows NT and Windows 95 have a lower intensity of use than in 1999. This shows that, albeit a slow change, IS departments do eventually change to newer operating systems.

The IS managers were also asked to specify the programming languages that are used in their department for the development of new applications, as well as for the maintenance of legacy systems. Table 4.4 contains a summary of the different programming languages used, and whether each language is used for new development or maintenance of legacy systems.

The most popular programming language in utilisation is Visual Basic with a 36% usage statistic for new development and close to 46% for maintenance. MSSQL was the second-most popular programming language together with Java. 18% of IS departments that were surveyed use each of the two aforementioned languages. When compared to the results obtained by Huisman (1999), it can be seen that Visual Basic was also the programming language of choice for a large amount of the IS departments at that time.

Programming Language	Kalanjee				Huisman (1999)			
	New Development		Maintenance		New Development		Maintenance	
	Number	%	Number	%	Number	%	Number	%
Visual Basic	4	36.4	5	45.5	34	47.2	29	40.3
MSSQL	2	18.2	2	18.2	0	0	0	0
Java	2	18.2	2	18.2	12	16.7	4	5.6
COBOL	1	9.1	2	18.2	13	18.1	25	34.7
Oracle	1	9.1	2	18.2	17	23.6	16	22.2
PROGRESS	1	9.1	1	9.1	3	4.2	3	4.2
Informix	1	9.1	1	9.1	3	4.2	4	5.6
Gupta	1	9.1	1	9.1	0	0	0	0
Delphi	1	9.1	1	9.1	9	12.5	8	11.1
Clarion	1	9.1	1	9.1	1	1.4	2	2.8
C++	1	9.1	1	9.1	15	20.8	12	16.7
C#	1	9.1	0	0	0	0	0	0
SAP	0	0	1	9.1	11	15.3	7	9.7
ICL VME Quickbuild	0	0	1	9.1	0	0	0	0

Table 4.4 Programming Languages used in the IS Departments

In Huisman's study, Oracle was used for both new development as well as maintenance. In this study it was found that Oracle is used mainly for the maintenance of existing systems. The use of COBOL has also appeared to have declined with only 9% of IS departments using it for development of new software and 18% for maintenance. This is compared to 18% for new development and close to 35% for maintenance in 1999.

By inspecting Table 4.4, it can be seen that the object oriented language Java and database related languages such as MSSQL and Oracle have maintained their status somewhat, while the use of older languages such as COBOL and PROGRESS has declined. This is possibly due to the fact that object-oriented languages are touted as being more effective in terms of re-use of code than the older languages. This ability to re-use code contributes to the sharing of responsibilities between developers.

On the whole, the results seem to indicate that South African organisations develop software on microcomputers running mainly Windows 98, UNIX, or Windows XP using a local area network. In addition to this, multiple programming languages are used for the development of new software and the maintenance of existing software.

As with Huisman's study, PC based environments, operating systems and programming languages seem to be the most prevalent.

4.2.3 Characteristics of the Systems Development Activities of IS Departments

IS department managers were asked to specify what percentage of there IS departments activities are allocated to the development of new applications, systems maintenance and support, and package customisation. These results are shown in Table 4.5.

Activity	Mean (n=11) %	0%	1-20%	21-40%	41-60%	61-80%	81- 100%
Development of new applications	34.1	1	2	5	2	1	0
Systems maintenance and support	53.0	1	0	2	5	3	0
Package customization	5.5	5	6	0	0	0	0

Table 4.5 IS Department Activities

Over half of the activities in the departments are devoted to the maintenance of existing systems and approximately 34% of the activities are allotted to the development of new applications. This result is similar to that which was obtained by Huisman (1999). Only 6% of the various IS departments' activities are allocated to package customisation. This contrasts with Huisman (1999), who found that almost 15% of IS departments customise packages. This difference could possibly be explained by the fact that all the organisations that outsourced their software development or purchased packages were not considered in this study.

The development activities of the IS departments was also investigated. Managers were asked to indicate what percentage of the IS departments development included systems planning, systems analysis, systems design, programming, testing, installation and evaluation. The information provided by the IS managers is shown in Table 4.6.

IS departments spend a third of their efforts on systems planning, analysis and design. Another third of their time is spent on programming, and testing, installation and evaluation are also allocated approximately a third of their time. Table 4.7 shows a comparison between this study and that of Huisman (1999).

Activity	Mean (n=11) %	0%	1-20%	21-40%	41-60%	61-80%	81- 100%
Systems planning	7.7	2	9	0	0	0	0
Systems analysis	10.5	1	10	0	0	0	0
Systems design	11.8	1	10	0	0	0	0
Programming	36.4	2	1	3	5	0	0
Testing	10.5	2	9	0	0	0	0
Installation	8.6	2	9	0	0	0	0
Evaluation	5.5	3	8	0	0	0	0
Other	8.2	10	0	0	0	0	1

Table 4.6 Development Activities

Activity	Kalanjee %	Huisman (1999) %
Systems planning	8	10
Systems analysis	10	13
Systems design	12	13
Programming	36	29
Testing	10	14
Installation	9	9
Evaluation	5	5

Table 4.7 Development Activities

The percentage of time spent on the various activities has not changed much and only a slight increase in the attention given to programming can be noted in the results of this study.

4.2.4 Characteristics of Systems Development Methodology Use

As explained in the definition of a systems development methodology, the focal point of this section will be on the use of systems development techniques, systems development process models and systems development methods which all form part of a systems development methodology.

The descriptive statistics for this section were conducted on the merged set of data, i.e. the manager and developer data were merged and the average of the individual responses was calculated.

- **Systems Development Techniques**

The list of techniques within the questionnaire were organised into four groups namely, techniques used during requirements analysis, techniques used during systems design, techniques used during testing and techniques used during maintenance. Respondents were asked to indicate, on a scale of 1 (nominal use) to 5 (extensive use), the extent to which each systems development technique was being used in the IS department. Table 4.8 shows the use of techniques in the IS departments ranked by intensity of use.

The highest use of techniques occurs in the requirements analysis phase where four of the techniques were used by all the organisations surveyed. A high use of techniques can also be seen in the testing and maintenance phases of systems development. Although the use of techniques is high, the intensity of use is not remarkable. Most of the techniques are used nominally with a majority of the values below average (< 3). This is in stark contrast to Huisman (1999) who recorded high use coupled with a correspondingly high intensity of use.

The techniques that did have a high intensity of use were techniques employed in the requirements analysis phase. Interestingly, rapid prototyping had a high intensity of use which indicates that there exists a requirement for applications to be developed quickly. Organisations need to gather requirements speedily and develop the application in the shortest time thereafter. This is due to the increased demand on organisations to deliver a working product as quickly as possible to clients or lose that client to another organisation that can deliver within the specified time-frame.

The generally low intensity of use across all the groups of techniques suggests that companies are shying away from using any organised technique and instead relying on ad-hoc techniques.

Techniques	Kalanjee			Huisman		
	Number (n=11)	%	Mean	Number (n=77)	%	Mean
<i>Requirements Analysis</i>						
Prototyping/ Rapid prototyping	9	81.8	3.70	55	71.4	3.17
Entity relationship diagrams	11	100.0	3.27	66	85.7	3.49
Data flow diagrams	11	100.0	3.14	65	84.4	3.46
Data normalization	11	100.0	3.14	68	88.3	3.66
Data dictionary	11	100.0	2.92	64	83.1	3.51
Flowcharts	10	90.9	2.90	67	87.0	3.27
Joint application development (JAD)	10	90.9	2.75	54	70.1	3.37
Process dependency diagrams	8	72.7	2.42	42	54.5	2.98
Process decomposition diagrams	9	81.8	2.30	40	51.9	3.10
Object-relationship models	9	81.8	2.20	41	53.2	3.07
Entity life cycle diagram	8	72.7	1.67	34	44.2	2.51
Object-behaviour models	8	72.7	1.58	25	32.5	2.89
Object-interaction models	8	72.7	1.54	30	39.0	3.03
<i>Design</i>						
Action diagrams	8	72.7	2.35	32	41.6	3.08
Process hierarchy diagrams	8	72.7	2.25	48	62.3	3.13
Transaction volume analysis	8	72.7	2.21	45	58.4	3.06
Decision tables/trees	8	72.7	2.06	42	54.5	3.15
Dialogue flow diagrams	8	72.7	2.04	23	29.9	2.76
Structure chart	8	72.7	1.90	35	45.5	3.19
Hierarchical input output diagrams	8	72.7	1.83	34	44.2	3.11
Class tables	8	72.7	1.56	31	40.3	2.99
Booch diagrams	8	72.7	1.04	11	14.3	2.14
Warnier-Orr diagrams	8	72.7	1.04	2	2.6	2.25
Nassi-Schneiderman diagrams	8	72.7	1.04	9	11.7	2.98
<i>Testing</i>						
Top-down testing	9	81.8	3.31	52	67.5	3.37
Walk-throughs	9	81.8	2.94	42	54.5	3.25
Bottom-up testing	9	81.8	2.67	41	53.2	3.13
White-box testing	9	81.8	1.80	22	28.6	3.26
Black-box testing	8	72.7	1.79	33	42.9	3.35
<i>Maintenance</i>						
Software change management procedures	10	90.9	3.22	60	77.9	3.75
Documentation change management procedures	9	81.8	2.85	61	79.2	3.03
Other	2	18.2	2.50	4	5.2	4.63
Configuration management procedures	9	81.8	2.44	51	66.2	3.18

Table 4.8 Techniques Used in IS Departments

- **Systems Development Process Models**

The process models that were listed in the questionnaire were based on those used by Huisman (1999). A 5-point scale was used and respondents were asked to indicate the extent to which their IS department's systems development practice utilised the listed process models. Respondents were allowed to mark more than one item. Table 4.9 shows the process models on which each of the organisation's systems development practices are based on.

Process Models	Kalanjee			Huisman (1999)	
	Number (n=11)	%	Mean	%	Mean
Evolutionary development in which the system is consciously planned to be delivered incrementally (by subsystems/ feature sets)	9	81.8	3.51	62	3.35
Prototyping using either paper prototypes (mock-ups) or computer-based prototypes.	9	81.8	3.33	58	2.99
Concurrent development in which the analysis, design and implementation take place virtually concurrently.	10	90.9	2.98	64	3.21
Phased process characterized by successive phases such as feasibility study, requirements phases such as feasibility study, requirements analysis, design, implementation, installation.	11	100.0	2.76	66	3.93
Spiral models characterized by successive spirals of analysis, design, prototype implementation, prototype use, evaluation.	9	81.8	1.93	47	2.88
Other	1	9.1	1.0	0	0

Table 4.9 Process Models Used

The phased process model is the most used, followed by the concurrent, evolutionary, prototyping and spiral models. Although the use of each model is almost evenly distributed, the extent to which each is used should be taken note of. Evolutionary and prototyping process models have above-average use with mean values >3.

The phased process model is reported as having a lower use as compared to 1999 and the use of the spiral model has also declined. Structured process models seem to have declined in popularity, and process models that allow multiple tasks to occur simultaneously, are used more frequently within organisations. The prevalence of prototyping in this list shows that clients want tangible evidence of the finished product before development is completed.

- **Systems Development Methods**

Respondents were provided with a list of systems development methods and asked to indicate, based on a 5-point scale, to what extent the listed methods were used in their respective IS departments. It was possible to select more than one method.

Table 4.10 shows the systems development methods used by the IS departments. The methods are grouped according to the approach on which they are based. The methods that were the most prominent were Rapid Application Development (RAD), Structured Systems Analysis and Design Method (SSADM) and Information Engineering (IE) with a use of approximately 72% each. The study conducted by Huisman (1999) revealed that RAD was the most popular method used; however, it did not have the highest intensity of use as it did in this study.

The prevalence of RAD ties in with the high intensity of use of the rapid prototyping technique discussed in the previous sections. This reiterates the need for hasty completion of systems in the software development industry.

The use of methodologies developed internally within organisations is more than a third less than what it was in 1999. This could possibly show that organisations have ceased developing methodologies that suit their specific needs and are instead doing development of software in a more unstructured or ad-hoc manner.

When considered in totality, the intensity of use of most methods has decreased significantly since the study by Huisman (1999) was conducted. Although the overall use is high, the intensity of use is relatively low which could be indicative of the fact that many organisations develop software on an ad-hoc basis and not by using any specific methodology.

Methods	Kalanjee			Huisman (1999)	
	Number (n=11)	%	Mean	%	Mean
<i>Structured Approach</i>					
SSADM (Structured Systems Analysis and Design Method)	8	72.7	1.88	31.2	3.30
SASD (Structured Analysis and Structured Design)	7	63.6	1.64	19.5	3.57
SSAD (Structured Systems Analysis and Design)	7	63.6	1.57	26.0	3.52
STRADIS (Structured Analysis, Design and Implementation of Information Systems)	7	63.6	1.14	39.0	3.64
YSM (Yourdon Systems Method)	7	63.6	1.00	5.2	2.50
Navigator	6	54.5	1.00	9.1	3.00
Method/1	6	54.5	1.00	7.8	3.08
<i>Information Modelling</i>					
IE (Information Engineering)	8	72.7	1.38	23.4	3.08
<i>Business Process-Oriented</i>					
Process Innovation (Business process re-engineering)	6	54.5	1.75	28.6	3.19
<i>Rapid Application Development</i>					
RAD (Rapid Application Development)	8	72.7	2.56	55.8	3.18
<i>Object-Oriented</i>					
OSA (Object-oriented Systems Analysis)	7	63.6	1.79	10.4	3.06
UML (Unified Modelling Language)	6	54.5	1.58	11.7	3.47
OMT (Object Modelling Technique by Rumbaugh)	6	54.5	1.17	10.4	3.06
OOA&D (Object Oriented Analysis and Design by Coad and Yourdon)	7	63.6	1.14	11.7	2.94
Booch	6	54.5	1.08	3.9	2.89
OOSE (Object Oriented Software Engineering by Jacobson)	6	54.5	1.00	6.5	2.20
OOA (Object Oriented Analysis by Shlaer and Mellor)	6	54.5	1.00	2.6	3.50
Fusion	6	54.5	1.00	1.3	4.00
MOSES	6	54.5	1.00	0.0	0.00
Objectory	6	54.5	1.00	3.9	3.58
<i>Socio-Technical Approach</i>					
ETHICS (Effective Technical and Human Implementation of Computer-based Systems)	6	54.5	1.00	2.6	3.50
<i>In-House</i>					
In-House	1	9.1	4.00	29.9	4.06
<i>Soft Systems Methodology</i>					
SSM (Soft Systems Methodology)	6	54.5	1.00	0.0	-
Multiview	6	54.5	1.00	0.0	-
<i>Other</i>					
ERwin	1	9.1	5.00	0	-
JSD (Jackson Systems Development)	6	54.5	1.67	7.8	2.92
SREM (Systems Requirements Engineering Methodology)	7	63.6	1.43	3.9	3.67
ISAC (Information Systems work and Analysis of Change)	6	54.5	1.00	0.0	-
Euromethod	6	54.5	1.00	1.3	2.00
KADS	6	54.5	1.00	0.0	-
Merise	6	54.5	1.00	0.0	-

Table 4.10 Systems Development Methods Used

4.3 DISCUSSION AND FINAL COMMENTS

The descriptive statistics that are presented in this chapter reveal some interesting facts about the profile of South African organisations and their use of methodologies. A vast majority of the organisations employ more than 200 people however, the size of the IS department is not proportionate to the size of the organisation.

The level of education of the developers is relatively low compared to when Huisman (1999) conducted her study. Notwithstanding this, developers have a high level of experience within the information technology industry and this could act as a compensating factor for their lack of education.

An important finding which must be mentioned is that the intensity of use of techniques and methods is very low compared to 1999. Although most organisations use some sort of technique or method, the degree of use is nominal. The low intensity of use supports the theory by Avison and Fitzgerald (2003) that the software development industry is undergoing a paradigm shift and moving into a post-methodology era where organisations are not using methodologies to develop software anymore. It also supports another theory that Fitzgerald *et al.* (2002) call Methods-in-Action. They suggest that the formalised methodologies that exist currently are not always used in the way they are meant to be used. Instead they are modified and even combined with other methodologies to create a new methodology that fulfils the needs of the organisation. Although not tested in this study, this theory could also be a contributing factor to the low intensity of use of techniques and methods.

Considering the results obtained from the descriptive statistics, it does seem as though organisations are moving away from the use of methodologies to develop software (Fitzgerald *et al.*, 2002). The statistics which follow in subsequent chapters will attempt to support this statement and provide conclusive proof that the use of systems development methodologies in South Africa has declined since Huisman's study in 1999.

CHAPTER 5

PERCEIVED MATURITY OF IS DEPARTMENTS AND THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

5.1 INTRODUCTION

In recent years, software projects have become more critical to the organisations that initiate them and it has become crucial for these projects to be completed successfully. In spite of this, software failures are still rife. Statistics released by The Standish Group (1995) show that 31.1% of software projects are cancelled before completion and that 52.7% of projects cost over 189% of their original estimates. These figures continue to escalate even though many organisations pour significant resources into software development methods such as CASE and Rapid Application Development (Jiang *et al.*, 2004).

The attention that the managerial side of a software development project requires is often underestimated and it is conducted with inadequate planning, poor understanding of the development process, and with the lack of a well-established management framework (Rai and Al-Hindi, 2000; Schwalbe, 2000). The Capability Maturity Model (CMM) is a framework that facilitates the evaluation and measurement of the managerial aspects of a software development project. It measures what is termed as the *maturity* of an organisation's software development processes (Paulk *et al.*, 1993).

This chapter will present the results of the investigation into the relationship between the perceived maturity of an IS department and the deployment of systems development methodologies. The concept of organisational software process maturity will be explained, followed by an overview of the Capability Maturity Model. The variables against which maturity is measured will also be introduced. The results of the investigation will then be presented and discussed.

5.2 **THE MATURITY OF IS DEPARTMENTS**

The extent to which a specific process is explicitly defined, managed, measured, controlled and effective is defined as the software process maturity of an organisation (Paulk *et al.* 1993). The Capability Maturity Model is an instrument that has been designed to measure the software process maturity of organisations. It consists of five levels of maturity with each level signifying the standard of the software process within the organisation.

The five levels comprising the CMM are discussed briefly below (Brandl, 2005).

- **Level 1 – Initial Level**

Organisations at the Initial Level have few or no rules for software development. During the duration of the project, they encounter crisis upon crisis and the ultimate success of the project depends on the developers' skills.

- **Level 2 – Repeatable**

At the Repeatable Level organisations are slightly better organised and software development is repeatable. There exist processes for requirements management, project planning, configuration management, software quality assurance, subcontract management, and project tracking. Project success is dependant on the skills of individual managers.

- **Level 3 – Defined**

Organisations at the Defined Level use the same process and methods across the entire organisation. There exist defined and documented processes which include all the Level 2 processes in addition to peer reviews, training programs, and formal cross-group coordination. Software development is repeatable and project success is dependant on training and documentation of processes.

- **Level 4 – Managed**

When the software process is at the Managed Level, organisations have management processes that measure process performance and adapt the development process for specific projects without comprising quality. Project problems are recognised swiftly and the appropriate corrective action is taken. The success of a project is dependant on the ability to objectively evaluate the project's performance, and have processes in place to take effective and early action in event of problems.

- **Level 5 – Optimising**

Organisations at the Optimising Level add continual process improvements to their existing Level 4 capabilities. Quantitative measures are used to proactively find the source of errors and correct processes. All projects are on target.

The CMM provides a reliable manner in which to measure the maturity of organisations software processes, however, it too has its share of criticisms (Wikipedia, 2005).

- The Capability Maturity Model has been unable to become a leading standard in the measurement of software process maturity.
- There exists no external body that certifies an organisation as CMM compliant. It is up to the organisation to perform an honest self-assessment.
- The CMM does not describe how to create an organisation that develops software effectively. It is merely comprised of behaviours and best practices that successful projects have demonstrated. In other words, being CMM compliant does not guarantee that a project will be successful. Instead, it only increases the chances of a project's success.

Although there is limited evidence on the success of the CMM, the available data does indicate that organisations adopting the CMM model tend to have substantially higher quality software, a faster cycle time, and higher development productivity (Butler, 1995; Dion, 1993; Humphrey *et al.*, 1991). Further evidence found by Herbsleb and Goldenson (1996) indicates that software development process maturity is associated with better organisational performance.

In recent years, the CMM has evolved into what is now known as Capability Maturity Model Integration (CMMI). CMMI is a way to assess and describe an organisation's software development process, compare it against industry standards, and help the organisation refine and improve the process (Kay, 2005). It combines a number of previously disparate elements and a chosen set of best practices. It is available in two flavours: staged and continuous. Staged CMMI is similar to the CMM as it has five levels which define the maturity of the software processes in an organisation. Continuous representation of CMMI allows an organisation to select specific improvements that best meet its business objectives and which will also minimise risk (Kay, 2005). Both of these types of CMMI yield the same results.

The instrument of measurement used in this study comprises of the CMM self-assessment questionnaire and algorithm that was utilised in the study conducted by Huisman (1999). This maintains the continuity between this study and that of Huisman's and allows reasonable comparisons to be made between the two studies. Although CMMI is the new standard for measuring the software process maturity of an organisation, the CMM level of the organisations' is investigated since the two frameworks are very similar to each other. Another reason is that this a comparative study which requires the same measure to be utilised in order to make reasonable comparisons.

5.3 THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

The instrument of measurement used in this study was proven to be reliable in Huisman's research conducted in 1999. The research variables used to define and measure deployment of systems development methodologies were thus retained and reused in this study.

The following three perspectives were used to measure the deployment of systems development methodologies (Huisman, 1999):

- The use of systems development methodologies
- The perceived support that systems development methodologies provide
- The perceived impact of systems development methodologies

These three perspectives will now be discussed in more detail.

5.3.1 The Use of Systems Development Methodologies

The use of systems development methodologies was measured by focusing on the horizontal and vertical use.

- ***Horizontal Use*** - Horizontal use relates to the methodology use across the entire organisation. This can be measured by the number of projects using the methodology, number of staff using the methodology and the number of projects per member of staff (Huisman, 1999). The average of two items, namely the proportion of projects that are developed in the IS department by applying systems development methodology knowledge, and the proportion of people in the IS department that apply systems development knowledge regularly, were used to calculate the horizontal use. This information was provided by the IS managers.
- ***Vertical Use*** - Vertical use relates to the components of the methodology which are being used and to what extent they are used. This is measured by usage of particular techniques of the methodology, usage of various stages of the methodology and usage of the systems development method (Huisman, 1999). Developers and managers were provided with a list of commercial systems development methods and asked to indicate to what extent their IS department uses them based on a scale from 1 (nominally) to 5 (extensively). They were also allowed to indicate what other commercial or in-house developed methods are being used by the IS department. Vertical use was calculated by merging the manager and developer data and then calculating the average of the responses which indicated that a specific method was used in a particular IS department. It is possible that an IS department can use more than one method and to facilitate the measurement of this, the maximum of all indicated methods was selected.

5.3.2 Perceived Methodology Support

Methodology support is measured by each of the following components (Huisman, 1999):

- **Perceived Support as Production Technology** – The support provided by the methodology for generating planning or design decisions, and subsequent developments arising from this planning. The perceived support as production technology was determined using the following three measures:
 - Support for organisational alignment
 - Support for technical design
 - Support for verification and validation
- **Perceived Support as Control Technology** – This is the support the methodology provides to the user for planning and enforcing rules that will govern the activities of team members during the planning and design process.
- **Perceived Support as Cognitive and Co-operative Technology** – This support allows the user to exchange information with other individuals in order to influence the concept, process or product of the planning and design team. The perceived support as cognitive and co-operative technology was determined using the following two measures:
 - Support for the common conception of SD practice
 - Support for the evaluation of SD practice

5.3.3 Perceived Impact of Systems Development Methodologies

Two aspects need to be considered in studying the impact of systems development methodologies (Huisman, 1999):

- The impact of the methodology on the quality of the product, i.e. the developed system.
- The impact of the methodology on the quality and productivity of the *process* of systems development. The perceived impact on the quality and productivity of the development process was determined using the following two measures:
 - Productivity effects and morale
 - Quality effects, goal achievement and reputation

5.4 RESULTS

5.4.1 Distribution of IS Departments across Maturity Levels

The IS department managers were asked for an indication of criteria relating to CMM that were fulfilled by their department. Twelve IS managers returned complete questionnaires for this section. Table 5.1 shows the percentage of key practices exercised at the different maturity levels by the IS departments.

Maturity Level	Mean (n=12) %	0%	1-20%	21-40%	41-60%	61-80%	81-100%
Level 2	42.4	0	0	3	8	1	0
Level 3	25.6	1	3	5	3	0	0
Level 4	12.5	3	7	1	1	0	0

Table 5.1 Percentage of Key Practices Exercised by IS Departments

On average, the organisations fulfilled 42.4% of the CMM level 2 criteria, 25.6% of the level 3 criteria, and 12.5% of the level 4 criteria. This suggests that the IS departments that participated in this study do not subscribe to the CMM's levels for determining software process maturity.

Table 5.2 shows the maturity levels and the distribution of the organisations across each level, and also includes the results of the study conducted by Huisman (1999) as comparison.

	Kalanjee		Huisman	
	N	%	N	%
Level 1 (Initial)	12	100.0	67	91.8
Level 2 (Repeatable)	0	0.0	4	5.5
Level 3 (Defined)	0	0.0	0	0.0
Level 4 (Managed)	0	0.0	2	2.7

Table 5.2 Distribution of IS Departments across Levels of Maturity

All of the departments surveyed are at the Initial level of the Capability Maturity Model. This is not startling since the findings of the descriptive statistics in Table 5.1 give a clear indication that there is nominal use of techniques and methods by the participating organisations.

Since CMM encapsulates the use of such methods, techniques and software development management tools during the software development process, it is almost expected that the IS departments would be at the initial level of maturity.

When compared to the results obtained by Huisman (1999), it is evident that the majority of organisations did not go beyond the Initial level of maturity in her study either. However, unlike in this study where all of the IS departments were at Level 1, Huisman's findings revealed that approximately 6% of organisations were at Level 2 and approximately 3% were at Level 4.

Cluster analysis was also conducted on the dataset. The K-means method of clustering was used to cluster the percentages of the key practices marked at the different maturity levels. The number of clusters was varied from two to five until an easily interpretable solution was found with three clusters. The results of the cluster analysis are shown in Table 5.3.

	Cluster 1 (n=4)	Cluster 2 (n=6)	Cluster 3 (n=2)
% of Level 2 Key Practices	39.6	37.5	62.5
% of Level 3 Key Practices	41.1	13.1	32.1
% of Level 4 Key Practices	8.3	5.6	41.7
Interpretation of Clusters (Maturity Stages)	Medium Maturity	Low Maturity	High Maturity

Table 5.3 Results of Cluster Analysis

The clusters approximately follow the hierarchy of levels so that a higher percentage of key practices are exercised at the lower level than at the higher levels. There are two deviations from this hierarchical structure. The first is in cluster 1, where the percentage of level 3 practices is higher than that of level 2 practices and the second is in cluster 3, where level 4 practices have a higher percentage than level 3 practices. This divergence confirms the findings made by Dekleva and Drehmer (1997) that organisations do not follow the CMM prescriptions verbatim. Instead, they skip key aspects of a particular maturity level and proceed with the practices of the next level when it is suitable to the organisation.

The distribution of the IS departments across the levels of maturity shows that all of the departments are at the Initial level. The cluster analysis shows that although this is the case, IS departments in Cluster 1 practice approximately 40% of the key practices of Levels 2 and 3. A peculiarity in the results is that organisations in Cluster 3 that were also categorised at the Initial level, practice over 60% of Level 2 key practices and over 40% of Level 4 key practices.

Table 5.4 shows the results from Huisman's cluster analysis. There are four clusters instead of three, but the distribution of organisations is comparable to that obtained in this study. There is a similar hierarchical pattern according to the percentage of key practices for each level, with a deviation only occurring in Cluster 4, where the percentage of Level 4 practices is higher than

that of Level 3 key practices. This is indicative of the fact that organisations skip key practices at a certain level instead of devoting their efforts to achieve that specific maturity level (Huisman, 1999).

	Cluster 1 (n=14)	Cluster 2 (n=17)	Cluster 3 (n=29)	Cluster 4 (n=9)
% of Level 2 Key Practices	15.5	65.7	56.3	84.3
% of Level 3 Key Practices	12.8	61.8	30.8	71.4
% of Level 4 Key Practices	3.6	39.2	20.7	82.4
Interpretation of Clusters (Maturity Stages)	Low Maturity	Medium Level 3	Medium Level 2	High Maturity

Table 5.4 Results of Cluster Analysis (Huisman, 1999)

Huisman's study also showed that a majority of the organisations were classified at the initial level. However, the cluster analysis revealed that many of the organisations that were classified at the initial level actually exercised a large percentage of the CMM key practices associated with the other maturity levels.

The results obtained from the cluster analysis are very similar to those obtained by Huisman (1999). Cluster 3 from this study corresponds to Cluster 4 in the study from 1999 given that both of the clusters have a higher percentage of Level 4 key practices than Level 3 key practices. The pattern thus deviates from the hierarchical pattern of the other clusters in both studies.

Huisman (1999) performed ANOVA/MANOVA analysis on the data however, after discussions with Statistical Consultation Services; it was recommended that ANOVA not be performed on this dataset as the sample population was small. Instead, simple regression was performed. No comparisons could therefore be drawn between this study and that of Huisman (1999) due to the fact that Huisman performed ANOVA/MANOVA analysis which is different to the simple regression performed in this study.

5.4.2 Regression Results

Simple regression was performed on the data to investigate the effect that the percentage of key practices performed at CMM maturity levels 2, 3, and 4, have on the deployment of systems development methods. The simple regression was performed with the percentage of key practices performed at CMM maturity levels 2, 3, and 4 as the independent variables, and the different measurements of deployment as the dependant variables.

The relationship between organisational maturity and the deployment of systems development methodologies is extrapolated on in this study by performing regression analyses that were not performed by Huisman (1999). Each of the factors that comprise a variable is tested to determine its relationship with the deployment of methodologies. This is done in an attempt to fulfil the need that Huisman (1999) identified for additional research into the relationship between organisational maturity and deployment of systems development methodologies.

A decision was taken to perform simple regression instead of multiple regression based on the advice received from Statistical Consultation Services. They also recommended that instead of using the *p-value* as a measure of statistical significance, the *Adjusted R²* value be used. The reasoning behind this was that the *p-value* assumes that a random sample population is used and that the population is large. In contrast, the *Adjusted R²* value is a statistical measure that is used to deliver comment on fixed samples that are not very large. The latter is the case for this study. The *Adjusted R²* value is significant when it is ≥ 0.25 .

The results are depicted in Table 5.5 and show that there is no relationship between deployment of methodologies and the percentage of key practices performed at CMM maturity Levels 2 and 4. There is however a strong relationship with deployment and the percentage of key practices performed at CMM maturity Level 3. The results in the table indicate that the percentage of key practices performed at CMM level 3 have an effect on the support for organisational deployment, support for technical design and support for verification and validation.

They also impact on the support for common conception of systems development practice, and the quality, goal achievement and reputation of the IS department.

Aspect of Deployment	Level 2	Level 3	Level 4
Support as Production Technology			
Support for Organisational Alignment	$R^2 = 0.12$ Adjusted $R^2 = -0.03$ $F = 0.79$ $p = 0.41$ $\beta = 0.34$	$R^2 = 0.50$ Adjusted $R^2 = 0.41^*$ $F = 5.87$ $p = 0.05$ $\beta = 0.70$	$R^2 = 0.09$ Adjusted $R^2 = -0.06$ $F = 0.58$ $p = 0.47$ $\beta = 0.30$
Support for Technical Design	$R^2 = 0.06$ Adjusted $R^2 = -0.13$ $F = 0.31$ $p = 0.60$ $\beta = 0.24$	$R^2 = 0.57$ Adjusted $R^2 = 0.48^*$ $F = 6.57$ $p = 0.05$ $\beta = 0.73$	$R^2 = 0.15$ Adjusted $R^2 = -0.02$ $F = 0.91$ $p = 0.38$ $\beta = 0.21$
Support for Verification and Validation	$R^2 = 0.20$ Adjusted $R^2 = 0.04$ $F = 1.25$ $p = 0.31$ $\beta = 0.45$	$R^2 = 0.55$ Adjusted $R^2 = 0.45^*$ $F = 6.03$ $p = 0.06$ $\beta = 0.75$	$R^2 = 0.05$ Adjusted $R^2 = -0.14$ $F = 0.24$ $p = 0.64$ $\beta = 0.39$
Support as Control Technology			
	$R^2 = 0.19$ Adjusted $R^2 = 0.05$ $F = 1.37$ $p = 0.29$ $\beta = 0.43$	$R^2 = 0.30$ Adjusted $R^2 = 0.18$ $F = 2.54$ $p = 0.16$ $\beta = 0.55$	$R^2 = 0.18$ Adjusted $R^2 = 0.04$ $F = 1.28$ $p = 0.30$ $\beta = 0.42$
Support as Cognitive and Co-operative Technology			
Support for Common Conception of SD Practice	$R^2 = 0.16$ Adjusted $R^2 = -0.00$ $F = 0.97$ $p = 0.37$ $\beta = 0.40$	$R^2 = 0.60$ Adjusted $R^2 = 0.52^*$ $F = 7.44$ $p = 0.04$ $\beta = 0.77$	$R^2 = 0.23$ Adjusted $R^2 = 0.08$ $F = 1.49$ $p = 0.28$ $\beta = 0.48$
Support for the Evaluation of SD Practice	$R^2 = 0.15$ Adjusted $R^2 = -0.03$ $F = 0.85$ $p = 0.40$ $\beta = 0.38$	$R^2 = 0.23$ Adjusted $R^2 = 0.08$ $F = 1.52$ $p = 0.27$ $\beta = 0.48$	$R^2 = 0.00$ Adjusted $R^2 = -0.19$ $F = 0.03$ $p = 0.88$ $\beta = 0.07$
Impact on the Developed System			
	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.00$ $p = 0.97$ $\beta = -0.01$	$R^2 = 0.3$ Adjusted $R^2 = 0.16$ $F = 2.15$ $p = 0.20$ $\beta = 0.55$	$R^2 = 0.00$ Adjusted $R^2 = -0.19$ $F = 0.03$ $p = 0.86$ $\beta = 0.08$
Impact on the Development Process			
Productivity Effects and Morale	$R^2 = 0.11$ Adjusted $R^2 = -0.07$ $F = 0.59$ $p = 0.48$ $\beta = -0.33$	$R^2 = 0.12$ Adjusted $R^2 = -0.06$ $F = 0.68$ $p = 0.45$ $\beta = 0.35$	$R^2 = 0.06$ Adjusted $R^2 = -0.13$ $F = 0.29$ $p = 0.61$ $\beta = -0.23$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.05$ Adjusted $R^2 = -0.15$ $F = 0.24$ $p = 0.64$ $\beta = 0.21$	$R^2 = 0.71$ Adjusted $R^2 = 0.66^*$ $F = 12.40$ $p = 0.02$ $\beta = 0.84$	$R^2 = 0.07$ Adjusted $R^2 = -0.11$ $F = 0.39$ $p = 0.56$ $\beta = 0.27$
Horizontal Use			
	$R^2 = 0.29$ Adjusted $R^2 = 0.17$ $F = 2.42$ $p = 0.17$ $\beta = 0.54$	$R^2 = 0.33$ Adjusted $R^2 = 0.22$ $F = 3.00$ $p = 0.13$ $\beta = 0.58$	$R^2 = 0.24$ Adjusted $R^2 = 0.12$ $F = 1.93$ $p = 0.21$ $\beta = 0.49$
Vertical Use			
	$R^2 = 0.02$ Adjusted $R^2 = -0.14$ $F = 0.14$ $p = 0.72$ $\beta = -0.15$	$R^2 = 0.18$ Adjusted $R^2 = 0.04$ $F = 1.28$ $p = 0.30$ $\beta = 0.41$	$R^2 = 0.02$ Adjusted $R^2 = -0.14$ $F = 0.13$ $p = 0.73$ $\beta = 0.14$

Table 5.5 Results of Simple Regression

5.5 DISCUSSION AND FINAL COMMENTS

The findings show that organisations approximately follow the hierarchy of levels as stipulated by the CMM. Although the organisations are all at the Initial CMM level, they do exercise key practices relating to Levels 2, 3 and 4 of the CMM. While the results of the cluster analysis show a similarity in the hierarchical structure of the clusters, there has been a decline in the application of the CMM key practices since Huisman's study in 1999. This could possibly be due to the fact that organisations have chosen to use other instruments to measure the maturity of their software process or that they are simply not perturbed about the level of maturity of the organisation's IS department. The sample in this study was smaller than the sample Huisman (1999) had and this could also have contributed to the results showing a decline in the application of CMM key practices.

The results of the simple regression shown in Table 5.5 support the findings made by Jiang *et al.* (2004) who also established that project performance is most related to the process engineering and organisational support activities of the CMM Level 3. The lack of significance of Level 2 and 4 key practices with that of deployment further supports the finding by Jiang *et al.* (2004) that the benefits of using the CMM may only be realised once Level 3 maturity is attained, and that the benefits may taper off after Level 3. As a consequence, this may cause organisations to hesitate before implementing the CMM practices since it can take months or even years before they gain value from the CMM. Additionally, after pouring many resources into the implementation of the CMM and eventually seeing benefits, it is possible that further benefits may tail off beyond CMM Level 3.

Another explanation for the inconsistent link between the CMM key practices and deployment of systems development methodologies could be that the organisations are intuitively applying the continuous representation of CMMI, which allows an organisation some flexibility to select specific improvements that best meet its business objectives and which also minimise risk.

The CMM does indeed appear to assist with the managerial aspects of software development, but considerable resources need to be spent before benefits can be seen. In today's cut-throat industry of software development, organisations need to seriously weigh the pros and cons of attaining CMM maturity level 4 and 5, since it may not be of any benefit to them.

CHAPTER 6

ORGANISATIONAL CULTURE AND THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

6.1 INTRODUCTION

Organisational culture is said to be a strategic resource that has value in ensuring the continued existence and success of organisations (Michalisin *et al.*, 1997; Hult *et al.*, 2002). This statement is supported by various studies that have linked organisational culture to broad strategic outcomes such as an organisation's ability to manage knowledge (Storck and Hill, 2000), innovation capability (Hauser, 1998), and strategic management of information technology (Kaarst-Brown and Robey, 1999; Reich and Benbasat, 2000; Schein, 1985).

The culture within an organisation can be studied in three ways (Robey and Azevedo, 1994):

- Visible symbols, artefacts, routines and practices
- Values and beliefs that can be articulated by informants
- Patterns of assumptions that are deeply held, possible without conscious awareness

Systems development methodologies thus form part of organisational culture since they can be considered as routines or practices within an organisation (Gallivan and Strite, 2005).

Consequently, the relationship between IT and organisational culture is a topic that has been researched since early field studies of IT implementation (Bostrom and Heinen, 1977; Markus, 1983). Researchers have identified problems with systems that work “right” technically, but are resisted by their users. This issue has been labelled as a lack of system/culture fit (Markus and Robey, 1983) and explained in terms of technologies having questionable “organisational validity” despite having adequate “technical validity” (Gallivan and Strite, 2005).

This chapter extrapolates on the abovementioned research by investigating the relationship between organisational culture and the deployment of systems development methodologies in particular. Prior research on this topic has been performed by Huisman (1999) and the results of this study will be compared to the results obtained by Huisman (1999). The Competing Values Framework (Quinn and Rohrbaugh, 1983, Quinn and Cameron, 1999; Quinn and Cameron,

1983; Woodman and Pasmore, 1991) is used to analyse the relationship between the organisational culture and the deployment of systems development methodologies.

Organisational culture and the Competing Values Framework will first be discussed, after which the variables against which organisational culture is measured are mentioned. The results obtained from the statistical analyses will then be presented and discussed.

6.2 ORGANISATIONAL CULTURE, THE COMPETING VALUES FRAMEWORK AND THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

6.2.1 Organisational Culture

Culture is reflected in the practices, values, beliefs and underlying assumptions of formal and informal groups (Quinn and Rohrbaugh, 1983; Schein, 1985). An organisation's culture is initially formed as a result of early experiences and the influence of early leaders. Over time, assumptions on how to operate become so implicitly embedded in the underlying assumptions of action, that they are difficult to articulate (Kaarst-Brown *et al.*, 2004)

Despite the ephemeral nature of organisational culture, newcomers become socialised, either directly through various artefacts such as the processes, rituals and structures of the organisation, or indirectly through espoused values and beliefs, language, and myths about past experiences (Louis, 1990).

Analysing cultural characteristics is challenging. According to Schein (1985), there are three interactive levels to culture:

- **Artefacts and Creations** – This is the most visible level of culture since it is the constructed physical and social environment of the organisation.
- **Values** – This level encompasses the “shoulds” or obligations of the organisation.
- **Basic Assumptions** – These are patterns that are implicitly taken for granted without being consciously aware of them.

In this study, the Competing Values Framework (Quinn and Rohrbaugh, 1983; Quinn and Kimberly, 1984) is applied to study the characteristics of organisational culture and how it influences the deployment of systems development methodologies.

6.2.2 The Competing Values Framework

This framework was originally developed for understanding the organisational effectiveness, competing tensions, and conflicts of three dimensions that are usually inherent in organisations (Al-Khalifa and Espinwall, 2001). The three dimensions are: emphasis on internal versus external, stability and control versus change and flexibility, and focal points of important organisational processes and outcomes (Zammuto and Krakower, 1991).

The Competing Values Framework (CVF) has been applied to a diverse set of issues and was extended by Quinn and Kimberly (1984) to examine organisational culture. This extended version of the CVF focuses on two dimensions visible in an organisation namely, the emphasis on internal versus external and stability versus control.

Based on these two dimensions, four types of organisational cultures emerge (Quinn and Kimberly, 1984; Huisman, 1999):

- **Group Culture** – This culture group is primarily focused on human relations and flexibility. Belonging, trust, and participation are its core values. Criteria which measure effectiveness include the development of human potential and member commitment.
- **Developmental Culture** – Emphasis is placed on the future and the scope of things to come. Effectiveness criteria emphasise insight, innovation, growth and adaptation to the external environment.
- **Rational Culture** – This culture group is very goal-driven and focuses on productivity, efficiency and profit from ventures.
- **Hierarchical Culture** – Rooted in stability, control and continuity, this culture group is oriented toward security, order and standardisation.

The four culture types are dominant and not mutually exclusive. This means that organisations are unlikely to comprise of only one culture but a mixture of the four and over time, one type of culture may emerge as the dominant one. Additionally, each of the culture types has a polar opposite. For example, group culture, which emphasises flexibility and internal focus, is

contrasted with rational culture, which stresses control and external focus. Similarly, developmental culture, which is characterised by flexibility and external focus, is in direct contrast to hierarchical culture, which places emphasis on control and internal focus (Denison and Spreitzer, 1991).

Huisman (1999) based the measurement of organisational culture on the Competing Values Framework. The same framework and instrument of measurement is used here to maintain the continuity between the two studies and provide a means to compare the results obtained from both the studies.

6.3 THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

The deployment of systems development methodologies and the three perspectives used to measure it were (Huisman, 1999):

- The use of systems development methodologies
- The perceived support that systems development methodologies provide
- The perceived impact of systems development methodologies

These three perspectives were discussed in detail in section 5.3 and thus only an overview of each is provided now.

6.3.1 The Use of Systems Development Methodologies

The use of systems development methodologies is measured by focusing on the horizontal and vertical use.

- ***Horizontal Use*** - Horizontal use relates to the methodology use across the entire organisation.
- ***Vertical Use*** - Vertical use relates to the components of the methodology which are being used and to what extent they are used.

6.3.2 Perceived Methodology Support

Methodology support is measured by each of the following components (Huisman, 1999):

- Perceived Support as Production Technology
- Perceived Support as Control Technology
- Perceived Support as Cognitive and Co-operative Technology

6.3.3 Perceived Impact of Systems Development Methodologies

Two aspects need to be considered in studying the impact of systems development methodologies (Huisman, 1999):

- The impact of the methodology on the quality of the product, i.e. the developed system.
- The impact of the methodology on the quality and productivity of the *process* of systems development.

6.4 RESULTS

6.4.1 Cluster Analysis

IS developers were asked to indicate the culture in their organisation. K-means clustering analysis with mean substitution was performed on the returned data. The number of clusters was varied from 2 to 5 and easily interpretable results were obtained with three and four clusters. The results are shown in Table 6.1 and Table 6.2, respectively.

	Cluster 1 (n=4)	Cluster 2 (n=4)	Cluster 3 (n=3)
Group	4.3	2.7	2.2
Developmental	3.7	2.2	2.6
Hierarchical	3.3	3.0	1.9
Rational	4.0	2.8	3.1
Interpretation of Clusters	Strong Comprehensive Culture	Weak Hierarchical Culture	Weak Rational Culture

Table 6.1 Results of Cluster Analysis (3 Clusters)

	Cluster 1 (n=2)	Cluster 2 (n=4)	Cluster 3 (n=2)	Cluster 4 (n=3)
Group	2.9	4.3	2.5	2.2
Developmental	2.3	3.7	2.0	2.6
Hierarchical	2.9	3.3	3.2	1.9
Rational	2.4	4.0	3.2	3.1
Interpretation of Clusters	Moderate Comprehensive Culture	Strong Comprehensive Culture	Weak Non-Developmental Culture	Weak Rational Culture

Table 6.2 Results of Cluster Analysis (4 Clusters)

The results obtained from the cluster analysis with three clusters show that four of the eleven organisations have a strong comprehensive culture. None of the four cultural groupings are dominant. The other two clusters contain organisations with weak cultural affiliations.

Cluster analysis with four clusters produced slightly different results. Two of the four clusters had moderate-to-strong cultural affiliations, while the remaining two were weakly associated to each of the four cultural groups.

When compared to the results obtained by Huisman (1999) shown in Table 6.3, it is apparent that organisations reported stronger cultural affiliations in 1999 than what they do now. Only one of the four clusters in her analysis had a weak cultural affiliation. The remaining three clusters contained organisations that were moderately-to-strongly associated with each of the different cultural groupings.

	Cluster 1 (n=26)	Cluster 2 (n=18)	Cluster 3 (n=13)	Cluster 4 (n=12)
Group	3.1	3.6	2.7	3.9
Developmental	2.9	3.4	2.2	3.8
Hierarchical	2.9	1.9	2.3	3.4
Rational	3.5	3.7	2.4	4.0
Interpretation of Clusters	Moderate Rational Culture	Moderate Non-Hierarchical Culture	Weak Group Culture	Strong Comprehensive Culture

Table 6.3 Results of Cluster Analysis (Huisman, 1999)

It is interesting to note that all of the organisations in Huisman's study have weak affiliations to hierarchical culture whereas none of the organisations in this study reported weak affiliations to the hierarchical culture group.

6.4.2 Regression Results

Simple linear regression was performed on the data to investigate the effect that various cultural groupings have on the deployment of systems development methodologies.

The independent variables used in this analysis were obtained by taking the mean of the items, which described each of the different cultural groups, for all of the organisations that participated in the survey. The different measurements of deployment were used as the dependant variables.

A decision was taken to perform simple regression instead of multiple regression based on the advice received from Statistical Consultation Services. They also recommended that instead of using the *p-value* as a measure of statistical significance, the *Adjusted R²* value be used. The reasoning behind this was that the *p-value* assumes that a random sample population is used and that the population is large. In contrast, the *Adjusted R²* value is a statistical measure that is used to deliver comment on fixed samples that are not very large. The latter is the case for this study. The *Adjusted R²* value is significant when it is ≥ 0.25 .

Comparisons are drawn with the results that were obtained by Huisman (1999). Although she performed multiple regression on the data, the two sets of results are comparable since the same instrument of measurement was used and the same dependant and independent variables were used in both the analyses.

The results of the regression analysis are shown in Table 6.4. In this study, the deployment of systems development methodologies is not related to group culture whatsoever.

When one considers the relationship between hierarchical culture and the deployment of SDMs, it is related to only one of the eleven measurements of deployment namely, support provided for the evaluation of systems development practice. According to the Competing Values Framework, hierarchical culture is associated with stability, control, and standardisation. The result obtained corroborates this fact since one can expect that an organisation that stresses the aforementioned values would use their SDM to evaluate their systems development practice thoroughly.

Aspect of Deployment	Group	Developmental	Rational	Hierarchical
Support as Production Technology				
Support for Organisational Alignment	$R^2 = 0.07$ Adjusted $R^2 = -0.09$ $F = 0.42$ $p = 0.54$ $\beta = 0.26$	$R^2 = 0.44$ Adjusted $R^2 = 0.33^*$ $F = 3.91$ $p = 0.10$ $\beta = 0.66$	$R^2 = 0.19$ Adjusted $R^2 = 0.03$ $F = 1.20$ $p = 0.32$ $\beta = 0.44$	$R^2 = 0.00$ Adjusted $R^2 = -0.16$ $F = 0.02$ $p = 0.90$ $\beta = 0.05$
Support for Technical Design	$R^2 = 0.22$ Adjusted $R^2 = 0.06$ $F = 1.37$ $p = 0.29$ $\beta = 0.46$	$R^2 = 0.76$ Adjusted $R^2 = 0.71^*$ $F = 15.47$ $p = 0.01$ $\beta = 0.87$	$R^2 = 0.49$ Adjusted $R^2 = 0.38^*$ $F = 4.72$ $p = 0.08$ $\beta = 0.70$	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.00$ $p = 0.95$ $\beta = 0.03$
Support for Verification and Validation	$R^2 = 0.21$ Adjusted $R^2 = 0.06$ $F = 1.36$ $p = 0.30$ $\beta = 0.46$	$R^2 = 0.59$ Adjusted $R^2 = 0.50^*$ $F = 7.06$ $p = 0.05$ $\beta = 0.77$	$R^2 = 0.52$ Adjusted $R^2 = 0.42^*$ $F = 5.32$ $p = 0.07$ $\beta = 0.72$	$R^2 = 0.08$ Adjusted $R^2 = -0.10$ $F = 0.46$ $p = 0.53$ $\beta = 0.29$
Support as Control Technology	$R^2 = 0.01$ Adjusted $R^2 = -0.16$ $F = 0.03$ $p = 0.87$ $\beta = 0.07$	$R^2 = 0.14$ Adjusted $R^2 = -0.04$ $F = 0.79$ $p = 0.41$ $\beta = 0.37$	$R^2 = 0.08$ Adjusted $R^2 = -0.11$ $F = 0.41$ $p = 0.55$ $\beta = 0.28$	$R^2 = 0.00$ Adjusted $R^2 = -0.17$ $F = 0.00$ $p = 0.95$ $\beta = -0.03$
Support as Cognitive and Co-operative Technology				
Support for Common Conception of SD Practice	$R^2 = 0.10$ Adjusted $R^2 = -0.08$ $F = 0.57$ $p = 0.48$ $\beta = 0.32$	$R^2 = 0.37$ Adjusted $R^2 = 0.25^*$ $F = 2.95$ $p = 0.15$ $\beta = 0.61$	$R^2 = 0.30$ Adjusted $R^2 = 0.15$ $F = 2.09$ $p = 0.21$ $\beta = 0.54$	$R^2 = 0.02$ Adjusted $R^2 = -0.18$ $F = 0.08$ $p = 0.79$ $\beta = 0.12$
Support for the Evaluation of SD Practice	$R^2 = 0.11$ Adjusted $R^2 = -0.07$ $F = 0.59$ $p = 0.48$ $\beta = 0.32$	$R^2 = 0.24$ Adjusted $R^2 = 0.09$ $F = 1.56$ $p = 0.27$ $\beta = 0.49$	$R^2 = 0.27$ Adjusted $R^2 = 0.12$ $F = 1.82$ $p = 0.24$ $\beta = 0.52$	$R^2 = 0.38$ Adjusted $R^2 = 0.26^*$ $F = 3.11$ $p = 0.14$ $\beta = 0.62$
Impact on the Developed System	$R^2 = 0.17$ Adjusted $R^2 = 0.00$ $F = 1.03$ $p = 0.36$ $\beta = 0.41$	$R^2 = 0.52$ Adjusted $R^2 = 0.43^*$ $F = 5.60$ $p = 0.06$ $\beta = 0.73$	$R^2 = 0.18$ Adjusted $R^2 = 0.01$ $F = 1.07$ $p = 0.35$ $\beta = 0.42$	$R^2 = 0.02$ Adjusted $R^2 = -0.18$ $F = 0.11$ $p = 0.76$ $\beta = -0.14$
Impact on the Development Process				
Productivity Effects and Morale	$R^2 = 0.22$ Adjusted $R^2 = 0.06$ $F = 1.40$ $p = 0.29$ $\beta = 0.47$	$R^2 = 0.55$ Adjusted $R^2 = 0.46^*$ $F = 6.11$ $p = 0.06$ $\beta = 0.74$	$R^2 = 0.13$ Adjusted $R^2 = -0.04$ $F = 0.75$ $p = 0.43$ $\beta = 0.36$	$R^2 = 0.08$ Adjusted $R^2 = -0.10$ $F = 0.46$ $p = 0.53$ $\beta = -0.29$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.09$ Adjusted $R^2 = -0.10$ $F = 0.48$ $p = 0.52$ $\beta = 0.30$	$R^2 = 0.58$ Adjusted $R^2 = 0.50^*$ $F = 7.00$ $p = 0.05$ $\beta = 0.76$	$R^2 = 0.27$ Adjusted $R^2 = 0.13$ $F = 1.87$ $p = 0.23$ $\beta = 0.52$	$R^2 = 0.01$ Adjusted $R^2 = -0.18$ $F = 0.07$ $p = 0.81$ $\beta = -0.11$
Horizontal Use	$R^2 = 0.00$ Adjusted $R^2 = -0.16$ $F = 0.02$ $p = 0.90$ $\beta = -0.05$	$R^2 = 0.30$ Adjusted $R^2 = 0.16^*$ $F = 2.13$ $p = 0.20$ $\beta = 0.55$	$R^2 = 0.30$ Adjusted $R^2 = 0.16$ $F = 2.11$ $p = 0.21$ $\beta = 0.54$	$R^2 = 0.01$ Adjusted $R^2 = -0.16$ $F = 0.04$ $p = 0.84$ $\beta = -0.08$
Vertical Use	$R^2 = 0.02$ Adjusted $R^2 = -0.15$ $F = 0.09$ $p = 0.77$ $\beta = -0.13$	$R^2 = 0.45$ Adjusted $R^2 = 0.33^*$ $F = 4.02$ $p = 0.10$ $\beta = 0.67$	$R^2 = 0.01$ Adjusted $R^2 = -0.19$ $F = 0.06$ $p = 0.82$ $\beta = 0.11$	$R^2 = 0.25$ Adjusted $R^2 = 0.12$ $F = 2.03$ $p = 0.20$ $\beta = 0.50$

Table 6.4 Results of Simple Regression

The results indicate that there is a relationship between rational culture and the support for technical design as well as the support for verification and validation. This is expected since rational culture is focused on productivity and efficiency.

The relationship between developmental culture and the deployment of systems development methodologies is the strongest of all the cultural orientations. It has an effect on every perspective of deployment besides support as control technology, support for the evaluation of SD practice and horizontal use. The lack of support for these aspects of deployment can be explained by the inherent nature of this organisational culture group. Developmental culture leans toward decentralisation, differentiation and flexibility. It thus seems logical that it would not advocate the use of any technology across the entire organisation (horizontal use) and also not render support as a control technology due to its characteristics of decentralisation and flexibility.

A comparison of the results obtained in this study and that of Huisman (1999) is shown in Table 6.5. The results obtained in this study are represented by a “K” and those obtained by Huisman (1999) are represented by an “H”. In stark contrast to the results obtained in this study, Huisman (1999) found that hierarchical culture influences the deployment of systems development methodologies the greatest. This is interesting since developmental culture and hierarchical culture are direct opposites of each other. Hierarchical culture emphasises control and internal focus, while developmental culture places emphasis on flexibility and external focus. Another point of distinction between the two culture groups is that, hierarchical culture slants towards longer time lines and centralisation and integration, whereas developmental culture is inclined towards decentralisation, differentiation and a competitive position of the overall system.

This difference in results between the two studies can possibly be explained by the increasingly competitive software development market, where organisations need to deliver software in the shortest time possible, or run the risk of losing clients to competitors.

Aspect of Deployment	Group	Developmental	Rational	Hierarchical
Support as Production Technology				
Support for Organisational Alignment		K		H
Support for Technical Design		K, H	K, H	H
Support for Verification and Validation		K, H	K	H
Support as Control Technology				H
Support as Cognitive and Co-operative Technology				
Support for Common Conception of SD Practice		K		H
Support for the Evaluation of SD Practice				K, H
Impact on the Developed System		K		H
Impact on the Development Process				
Productivity Effects and Morale		K, H	H	
Quality Effects, Goal Achievement and Reputation		K	H	H
Horizontal Use				
Vertical Use		K	H	H

Table 6.5 Results of Multiple Regression (Huisman, 1999)

6.5 DISCUSSION AND FINAL COMMENTS

The results of the cluster analysis show that organisations reported a generally weaker affiliation to particular cultural groups than they did in 1999. There has been a dramatic shift in the cultural orientation of organisations' since the study by Huisman was conducted in 1999. This shift is possibly due to the fact that the *types* of organisations that have entered the information systems development arena are different to those in the past and that they also have a stronger affiliation to the developmental culture.

The regression analysis performed on the data reveal some highly interesting results. Both studies show that group culture has no bearing on the deployment of systems development methodologies. Developmental culture has the greatest influence on the deployment of systems development methodologies.

This is in contrast to Huisman (1999) who found that hierarchical culture, which is the opposite of developmental culture, influences the deployment of systems development methodologies the most.

The fact that the type of organisational culture which most influences deployment of methodologies is the antithesis of what it was in 1999 is a point of interest. It suggests that since 1999, organisations with a different culture orientation are deploying SDMs more successfully. Developmental culture does lean toward decentralization and growth. It also has an external focus and encourages insight, innovation and adaptation. The globalisation of the software development market in recent years could be a reason for the strong relationship between developmental culture and the deployment of systems development methodologies.

The characteristics of the developmental culture do encourage globalisation and the results obtained in this study seem to corroborate this fact. However, additional research into this topic is necessary to determine exactly what significance the globalisation of the software development market has on organisational culture.

CHAPTER 7

THE ORGANISATIONAL DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

7.1 INTRODUCTION

This chapter will introduce and discuss the results of the statistical analyses conducted to determine which organisational factors influence the deployment of systems development methodologies. This is a topic which needs to be investigated since researchers (Fichman, 1992; Senn and Wynekoop, 1995; Lai and Guynes, 1997) have found that the implementation of a technology at the organisational level differs from that at individual level. The investigation into the deployment of systems development methodologies at the individual level will follow in the next chapter.

Since this is a comparative study with that of Huisman (1999) and the instrument of measurement was proven to be reliable in her study, the organisational factors used in all analyses are identical to those defined and used by her.

A description of the organisational factors will be provided and an overview of the elements that comprise the deployment of methodologies will be given. The results will then be presented and compared to those of Huisman (1999).

7.2 ORGANISATIONAL CHARACTERISTICS AND THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

The organisational and task characteristics considered in this analysis are shown in Table 7.1. The factors were selected according to the conceptual research model which was defined and tested by Huisman (1999) and discussed in detail in section 3.2. The measurement of the factors was done with the instrument of measurement which was tested for reliability and used by Huisman (1999).

Characteristics	Factors
Organisational	Business Area IS Department Size Maturity Culture
Task	Time Spent on the Development of New Applications Time Spent on the Planning, Analysis and Design of Systems

Table 7.1 Factors Studied to Determine Effect on Organisational Deployment of Systems Development Methodologies

7.2.1 Organisational Characteristics

- ***Business Area***

Managers were asked to indicate the core business area of their organisation. They were presented with the following list to choose from: Administrative Services, Finance/Banking/Insurance/, Software House/Software Consulting, Manufacturing, Retail/Wholesale, Education, and Other.

To facilitate the analysis of the influence of business area on the deployment of systems development methodologies, the business area variable had to be changed from a categorical to an ordinal variable. This was performed by Huisman (1999) after conferring with Statistical Consultation Services at the university during the course of her research. It was thus not deemed necessary to consult them again on this issue and the procedure suggested by them was used again in this study.

ANOVA/MANOVA analysis was performed to determine whether the use of systems development methodologies differs significantly between the IS departments in the different business areas. Three organisations indicated their business area as “Other”. These cases were considered individually and an attempt was made to place them into one of the listed categories. The ANOVA analysis results shown in Table 7.2 reveal that there are no significant differences in the use of methodologies between the various business areas. The business area is thus omitted for the purposes of this analysis.

	Horizontal Methodology Use	Vertical Methodology Use
Administrative Services	4.00	5.00
Software House/Software Consulting	2.67	4.33
Manufacturing	3.33	3.00
Adjusted R ²	-0.25	0.13
p-value	0.75	0.30

Table 7.2 Results of ANOVA Analysis

- ***IS Department Size***

IS managers were asked to indicate the total number of people employed by there IS department at all locations.

- ***Maturity of IS Department***

The maturity of the IS departments was defined according to the Capability Maturity Model and were measured and discussed in detail in section 5.2. The majority of organisations were categorised under the Initial level of maturity. The results of the cluster analysis that was performed using the percentages of the key practices as clustering variables are used in the analysis. The different clusters that describe the maturity stages of each of the departments are:

- Low Maturity
- Medium Maturity
- High Maturity

- ***Organisational Culture***

The organisational culture was measured and discussed thoroughly in section 6.2. The four indicators of culture analysed were:

- Group culture
- Developmental culture
- Hierarchical culture
- Rational culture

Although the effect that organisational culture has on the deployment of systems development methodologies was already reported on in section 6.4.2, it is repeated here for the purpose of providing a comprehensive view of the impact that organisational characteristics have on the deployment of systems development methodologies.

7.2.2 Task Characteristics

Task characteristics were measured by asking managers to indicate the percentage of effort their IS department dedicated to the development of new applications and to systems planning, analysis and design.

7.2.3 Deployment of Systems Development Methodologies

The deployment of systems development methodologies is studied using the three perspectives described below (Huisman, 1999).

- The use of systems development methodologies
- The perceived support that systems development methodologies provide
- The perceived impact of systems development methodologies

These three perspectives were discussed in detail in section 5.3 and thus only an overview of each is provided now.

- ***Use of Systems Development Methodologies***

The use of systems development methodologies is measured by focusing on horizontal and vertical use.

- *Horizontal Use* - Horizontal use relates to the frequency of methodology use across the IS department.
- *Vertical Use* - Vertical use relates to the components of the methodology which are being used and the intensity of their use.

- ***Perceived Methodology Support***

Methodology support is measured by each of the following components (Huisman, 1999):

- Perceived Support as Production Technology
- Perceived Support as Control Technology
- Perceived Support as Cognitive and Co-operative Technology

- ***Perceived Impact of Systems Development Methodologies***

Two aspects need to be considered in studying the impact of systems development methodologies (Huisman, 1999):

- The impact of the methodology on the quality of the product, i.e. the developed system.
- The impact of the methodology on the quality and productivity of the *process* of systems development.

7.3 RESULTS

Simple linear regression was performed on the data to investigate the effect that organisational characteristics have on the deployment of systems development methodologies. A decision was taken to perform simple regression instead of multiple regression based on the advice received from Statistical Consultation Services. They also recommended that instead of using the *p-value* as a measure of statistical significance, the *Adjusted R²* value be used.

The reasoning behind this was that the *p-value* assumes that a random sample population is used that the population is large. In contrast, the *Adjusted R²* value is a statistical measure that is used to deliver comment on fixed samples that are not very large. The latter is the case for this study. The *Adjusted R²* value is significant when it is ≥ 0.25 .

The independent variables used in this analysis were obtained by taking the mean of the items in the questionnaire that described each of the factors which comprise organisational and task characteristics for each of the organisations. The different measurements of deployment were used as the dependant variables.

Table 7.3 shows the effect that IS department size, maturity, the time spent on planning, analysis and design, and the time spent on new development have on the deployment of systems development methodologies.

IS department size has a significant negative effect on the deployment of systems development methodologies. Maturity and the time spent on the development of new applications were not found to be highly influential on the deployment of methodologies. There is no relation whatsoever between the time spent on planning, analysing and designing the system, and the deployment of methodologies.

The impact of organisational culture on the deployment of systems development methodologies is depicted in Table 7.4. Table 7.5 shows a comparison of the results obtained from the regression performed in this study and that of Huisman (1999). These results were discussed in detail in section 6.4.2.

Aspect of Deployment	IS Department Size	Maturity	Time: Planning, Analysis, Design	Time: New Development
Support as Production Technology	$R^2 = 0.36$ Adjusted $R^2 = 0.25^*$ $F = 3.38$ $p = 0.12$ $\beta = -0.60$	$R^2 = 0.24$ Adjusted $R^2 = 0.12$ $F = 1.92$ $p = 0.22$ $\beta = 0.50$	$R^2 = 0.03$ Adjusted $R^2 = -0.14$ $F = 0.16$ $p = 0.70$ $\beta = 0.16$	$R^2 = 0.02$ Adjusted $R^2 = -0.17$ $F = 0.13$ $p = 0.74$ $\beta = -0.16$
Support for Organisational Alignment	$R^2 = 0.21$ Adjusted $R^2 = 0.08$ $F = 1.63$ $p = 0.25$ $\beta = -0.46$	$R^2 = 0.36$ Adjusted $R^2 = 0.25^*$ $F = 3.35$ $p = 0.12$ $\beta = 0.60$	$R^2 = 0.09$ Adjusted $R^2 = -0.06$ $F = 0.62$ $p = 0.46$ $\beta = 0.31$	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.00$ $p = 0.99$ $\beta = -0.01$
Support for Technical Design	$R^2 = 0.48$ Adjusted $R^2 = 0.37^*$ $F = 4.56$ $p = 0.09$ $\beta = -0.69$	$R^2 = 0.28$ Adjusted $R^2 = 0.14$ $F = 1.99$ $p = 0.22$ $\beta = 0.53$	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.00$ $p = 0.96$ $\beta = 0.06$	$R^2 = 0.11$ Adjusted $R^2 = -0.11$ $F = 0.49$ $p = 0.52$ $\beta = -0.33$
Support for Verification and Validation	$R^2 = 0.34$ Adjusted $R^2 = 0.21$ $F = 2.57$ $p = 0.17$ $\beta = -0.58$	$R^2 = 0.37$ Adjusted $R^2 = 0.24$ $F = 2.91$ $p = 0.15$ $\beta = 0.61$	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.02$ $p = 0.90$ $\beta = 0.53$	$R^2 = 0.00$ Adjusted $R^2 = -0.25$ $F = 0.00$ $p = 0.94$ $\beta = -0.04$
Support as Control Technology	$R^2 = 0.27$ Adjusted $R^2 = 0.15$ $F = 2.19$ $p = 0.19$ $\beta = -0.52$	$R^2 = 0.23$ Adjusted $R^2 = 0.10$ $F = 1.76$ $p = 0.23$ $\beta = 0.48$	$R^2 = 0.07$ Adjusted $R^2 = -0.09$ $F = 0.43$ $p = 0.54$ $\beta = 0.26$	$R^2 = 0.10$ Adjusted $R^2 = -0.07$ $F = 0.58$ $p = 0.48$ $\beta = -0.32$
Support as Cognitive and Co-operative Technology	$R^2 = 0.68$ Adjusted $R^2 = 0.35^*$ $F = 4.20$ $p = 0.10$ $\beta = -0.68$	$R^2 = 0.26$ Adjusted $R^2 = 0.11$ $F = 1.76$ $p = 0.24$ $\beta = 0.51$	$R^2 = 0.01$ Adjusted $R^2 = -0.19$ $F = 0.04$ $p = 0.85$ $\beta = -0.09$	$R^2 = 0.04$ Adjusted $R^2 = -0.20$ $F = 0.17$ $p = 0.70$ $\beta = -0.20$
Support for Common Conception of SD Practice	$R^2 = 0.49$ Adjusted $R^2 = 0.38^*$ $F = 4.73$ $p = 0.08$ $\beta = -0.70$	$R^2 = 0.24$ Adjusted $R^2 = 0.09$ $F = 1.58$ $p = 0.26$ $\beta = 0.49$	$R^2 = 0.01$ Adjusted $R^2 = 0.18$ $F = 0.07$ $p = 0.80$ $\beta = -0.12$	$R^2 = 0.08$ Adjusted $R^2 = -0.15$ $F = 0.35$ $p = 0.58$ $\beta = -0.28$
Support for the Evaluation of SD Practice	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.02$ $p = 0.90$ $\beta = -0.06$	$R^2 = 0.12$ Adjusted $R^2 = -0.05$ $F = 0.70$ $p = 0.44$ $\beta = 0.35$	$R^2 = 0.10$ Adjusted $R^2 = -0.08$ $F = 0.54$ $p = 0.50$ $\beta = 0.31$	$R^2 = 0.29$ Adjusted $R^2 = 0.12$ $F = 1.66$ $p = 0.27$ $\beta = 0.54$
Impact on the Developed System	$R^2 = 0.47$ Adjusted $R^2 = 0.36^*$ $F = 4.39$ $p = 0.09$ $\beta = -0.68$	$R^2 = 0.18$ Adjusted $R^2 = 0.01$ $F = 1.06$ $p = 0.35$ $\beta = 0.42$	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.00$ $p = 0.98$ $\beta = -0.01$	$R^2 = 0.52$ Adjusted $R^2 = 0.40^*$ $F = 4.30$ $p = 0.11$ $\beta = -0.72$
Impact on the Development Process	$R^2 = 0.59$ Adjusted $R^2 = 0.51^*$ $F = 7.20$ $p = 0.04$ $\beta = -0.77$	$R^2 = 0.10$ Adjusted $R^2 = -0.08^*$ $F = 0.56$ $p = 0.49$ $\beta = 0.32$	$R^2 = 0.02$ Adjusted $R^2 = -0.17$ $F = 0.11$ $p = 0.76$ $\beta = -0.14$	$R^2 = 0.51$ Adjusted $R^2 = 0.38^*$ $F = 4.08$ $p = 0.11$ $\beta = -0.71$
Productivity Effects and Morale	$R^2 = 0.48$ Adjusted $R^2 = 0.38^*$ $F = 4.70$ $p = 0.08$ $\beta = -0.70$	$R^2 = 0.03$ Adjusted $R^2 = -0.17$ $F = 0.14$ $p = 0.72$ $\beta = 0.17$	$R^2 = 0.02$ Adjusted $R^2 = -0.17$ $F = 0.13$ $p = 0.74$ $\beta = 0.16$	$R^2 = 0.48$ Adjusted $R^2 = 0.35^*$ $F = 3.65$ $p = 0.13$ $\beta = -0.69$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.55$ Adjusted $R^2 = 0.46^*$ $F = 6.21$ $p = 0.05$ $\beta = -0.74$	$R^2 = 0.22$ Adjusted $R^2 = 0.06$ $F = 1.39$ $p = 0.29$ $\beta = 0.47$	$R^2 = 0.01$ Adjusted $R^2 = -0.19$ $F = 0.06$ $p = 0.82$ $\beta = -0.11$	$R^2 = 0.20$ Adjusted $R^2 = 0.00$ $F = 1.00$ $p = 0.37$ $\beta = -0.45$
Horizontal Use	$R^2 = 0.06$ Adjusted $R^2 = -0.07$ $F = 0.46$ $p = 0.52$ $\beta = 0.25$	$R^2 = 0.70$ Adjusted $R^2 = 0.65^*$ $F = 13.87$ $p = 0.01$ $\beta = 0.84$	$R^2 = 0.13$ Adjusted $R^2 = -0.01$ $F = 0.90$ $p = 0.38$ $\beta = 0.36$	$R^2 = 0.09$ Adjusted $R^2 = -0.09$ $F = 0.49$ $p = 0.51$ $\beta = 0.30$
Vertical Use	$R^2 = 0.02$ Adjusted $R^2 = -0.12$ $F = 0.16$ $p = 0.70$ $\beta = 0.15$	$R^2 = 0.13$ Adjusted $R^2 = -0.01$ $F = 0.91$ $p = 0.38$ $\beta = 0.36$	$R^2 = 0.04$ Adjusted $R^2 = -0.11$ $F = 0.28$ $p = 0.61$ $\beta = -0.21$	$R^2 = 0.01$ Adjusted $R^2 = -0.19$ $F = 0.06$ $p = 0.82$ $\beta = -0.11$

Table 7.3 Impact of IS Department Size, Maturity and Time Factors on the Deployment of Systems Development Methodologies

Aspect of Deployment	Group	Developmental	Rational	Hierarchical
Support as Production Technology				
Support for Organisational Alignment	$R^2 = 0.07$ Adjusted $R^2 = -0.09$ $F = 0.42$ $p = 0.54$ $\beta = 0.26$	$R^2 = 0.44$ Adjusted $R^2 = 0.33^*$ $F = 3.91$ $p = 0.10$ $\beta = 0.66$	$R^2 = 0.19$ Adjusted $R^2 = 0.03$ $F = 1.20$ $p = 0.32$ $\beta = 0.44$	$R^2 = 0.00$ Adjusted $R^2 = -0.16$ $F = 0.02$ $p = 0.90$ $\beta = 0.05$
Support for Technical Design	$R^2 = 0.22$ Adjusted $R^2 = 0.06$ $F = 1.37$ $p = 0.29$ $\beta = 0.46$	$R^2 = 0.76$ Adjusted $R^2 = 0.71^*$ $F = 15.47$ $p = 0.01$ $\beta = 0.87$	$R^2 = 0.49$ Adjusted $R^2 = 0.38^*$ $F = 4.72$ $p = 0.08$ $\beta = 0.70$	$R^2 = 0.00$ Adjusted $R^2 = -0.20$ $F = 0.00$ $p = 0.95$ $\beta = 0.03$
Support for Verification and Validation	$R^2 = 0.21$ Adjusted $R^2 = 0.06$ $F = 1.36$ $p = 0.30$ $\beta = 0.46$	$R^2 = 0.59$ Adjusted $R^2 = 0.50^*$ $F = 7.06$ $p = 0.05$ $\beta = 0.77$	$R^2 = 0.52$ Adjusted $R^2 = 0.42^*$ $F = 5.32$ $p = 0.07$ $\beta = 0.72$	$R^2 = 0.08$ Adjusted $R^2 = -0.10$ $F = 0.46$ $p = 0.53$ $\beta = 0.29$
Support as Control Technology	$R^2 = 0.01$ Adjusted $R^2 = -0.16$ $F = 0.03$ $p = 0.87$ $\beta = 0.07$	$R^2 = 0.14$ Adjusted $R^2 = -0.04$ $F = 0.79$ $p = 0.41$ $\beta = 0.37$	$R^2 = 0.08$ Adjusted $R^2 = -0.11$ $F = 0.41$ $p = 0.55$ $\beta = 0.28$	$R^2 = 0.00$ Adjusted $R^2 = -0.17$ $F = 0.00$ $p = 0.95$ $\beta = -0.03$
Support as Cognitive and Co-operative Technology				
Support for Common Conception of SD Practice	$R^2 = 0.10$ Adjusted $R^2 = -0.08$ $F = 0.57$ $p = 0.48$ $\beta = 0.32$	$R^2 = 0.37$ Adjusted $R^2 = 0.25^*$ $F = 2.95$ $p = 0.15$ $\beta = 0.61$	$R^2 = 0.30$ Adjusted $R^2 = 0.15$ $F = 2.09$ $p = 0.21$ $\beta = 0.54$	$R^2 = 0.02$ Adjusted $R^2 = -0.18$ $F = 0.08$ $p = 0.79$ $\beta = 0.12$
Support for the Evaluation of SD Practice	$R^2 = 0.11$ Adjusted $R^2 = -0.07$ $F = 0.59$ $p = 0.48$ $\beta = 0.32$	$R^2 = 0.24$ Adjusted $R^2 = 0.09$ $F = 1.56$ $p = 0.27$ $\beta = 0.49$	$R^2 = 0.27$ Adjusted $R^2 = 0.12$ $F = 1.82$ $p = 0.24$ $\beta = 0.52$	$R^2 = 0.38$ Adjusted $R^2 = 0.26^*$ $F = 3.11$ $p = 0.14$ $\beta = 0.62$
Impact on the Developed System	$R^2 = 0.17$ Adjusted $R^2 = 0.00$ $F = 1.03$ $p = 0.36$ $\beta = 0.41$	$R^2 = 0.52$ Adjusted $R^2 = 0.43^*$ $F = 5.60$ $p = 0.06$ $\beta = 0.73$	$R^2 = 0.18$ Adjusted $R^2 = 0.01$ $F = 1.07$ $p = 0.35$ $\beta = 0.42$	$R^2 = 0.02$ Adjusted $R^2 = -0.18$ $F = 0.11$ $p = 0.76$ $\beta = -0.14$
Impact on the Development Process				
Productivity Effects and Morale	$R^2 = 0.22$ Adjusted $R^2 = 0.06$ $F = 1.40$ $p = 0.29$ $\beta = 0.47$	$R^2 = 0.55$ Adjusted $R^2 = 0.46^*$ $F = 6.11$ $p = 0.06$ $\beta = 0.74$	$R^2 = 0.13$ Adjusted $R^2 = -0.04$ $F = 0.75$ $p = 0.43$ $\beta = 0.36$	$R^2 = 0.08$ Adjusted $R^2 = -0.10$ $F = 0.46$ $p = 0.53$ $\beta = -0.29$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.09$ Adjusted $R^2 = -0.10$ $F = 0.48$ $p = 0.52$ $\beta = 0.30$	$R^2 = 0.58$ Adjusted $R^2 = 0.50^*$ $F = 7.00$ $p = 0.05$ $\beta = 0.76$	$R^2 = 0.27$ Adjusted $R^2 = 0.13$ $F = 1.87$ $p = 0.23$ $\beta = 0.52$	$R^2 = 0.01$ Adjusted $R^2 = -0.18$ $F = 0.07$ $p = 0.81$ $\beta = -0.11$
Horizontal Use	$R^2 = 0.00$ Adjusted $R^2 = -0.16$ $F = 0.02$ $P = 0.90$ $\beta = -0.05$	$R^2 = 0.30$ Adjusted $R^2 = 0.16^*$ $F = 2.13$ $p = 0.20$ $\beta = 0.55$	$R^2 = 0.30$ Adjusted $R^2 = 0.16$ $F = 2.11$ $p = 0.21$ $\beta = 0.54$	$R^2 = 0.01$ Adjusted $R^2 = -0.16$ $F = 0.04$ $p = 0.84$ $\beta = -0.08$
Vertical Use	$R^2 = 0.02$ Adjusted $R^2 = -0.15$ $F = 0.09$ $P = 0.77$ $\beta = -0.13$	$R^2 = 0.45$ Adjusted $R^2 = 0.33^*$ $F = 4.02$ $p = 0.10$ $\beta = 0.67$	$R^2 = 0.01$ Adjusted $R^2 = -0.19$ $F = 0.06$ $p = 0.82$ $\beta = 0.11$	$R^2 = 0.25$ Adjusted $R^2 = 0.12$ $F = 2.03$ $p = 0.20$ $\beta = 0.50$

Table 7.4 Impact of Culture on the Deployment of Systems Development Methodologies

Aspect of Deployment	Group	Developmental	Rational	Hierarchical
Support as Production Technology				
Support for Organisational Alignment		K		H
Support for Technical Design		K, H	K, H	H
Support for Verification and Validation		K, H	K	H
Support as Control Technology				H
Support as Cognitive and Co-operative Technology				
Support for Common Conception of SD Practice		K		H
Support for the Evaluation of SD Practice				K, H
Impact on the Developed System		K		H
Impact on the Development Process				
Productivity Effects and Morale		K, H	H	
Quality Effects, Goal Achievement and Reputation		K	H	H
Horizontal Use				
Vertical Use		K	H	H

Table 7.5 Comparison of Results of Regression with Culture Group as Independent Variable between this study and Huisman (1999)

Table 7.6 compares the results obtained in this study to those obtained by Huisman (1999). The multiple regression performed by Huisman was on the actual variables and not their underlying factors. This means that the independent variables were not tested against the individual elements such as support for technical design, support for organisational alignment and support for verification and validation, which all constitute support as production technology. Instead, each of the independent variables was tested against support as production technology as a whole.

In addition to the regression performed on the individual elements of each variable that describe deployment, regression was also performed on the actual variable to facilitate the accurate comparison of results between this study and Huisman (1999). This is reflected in Table 7.6. The results obtained in this study are represented by a “K” and those obtained by Huisman (1999) are represented by an “H”.

Aspect of Deployment	IS Department Size	Maturity	Time: Planning, Analysis, Design	Time: New Development
Support as Production Technology	K, H			H
Support for Organisational Alignment		K		
Support for Technical Design	K			
Support for Verification and Validation				
Support as Control Technology	H			
Support as Cognitive and Co-operative Technology	K			H
Support for Common Conception of SD Practice	K			
Support for the Evaluation of SD Practice				
Impact on the Developed System	K, H			K
Impact on the Development Process	K			K
Productivity Effects and Morale	K			K
Quality Effects, Goal Achievement and Reputation	K			
Horizontal Use		K, H		H
Vertical Use		H		

Table 7.6 Comparison of Results between Huisman and Iivari (2002) and this study

7.4 DISCUSSION AND FINAL COMMENTS

IS department size has a significant negative effect on the deployment of systems development methodologies. In particular, it influences the support aspects of deployment and it also has an effect on the development process and the developed system. The negative impact is surprising since it is often perceived that large organisations implement systems development methodologies more readily than their smaller counterparts. It is often assumed that the larger the IS department, the greater the support available for processes and tasks associated with software development to be implemented and completed. The results of this study show that this is not always the case.

Table 7.3 also shows the impact of an organisation's maturity stage on the deployment of methodologies. The support for organisational alignment and horizontal use are impacted by maturity stage. According to the CMM the more processes, techniques, and methods that an organisation has in place to manage its software development, the higher its maturity level.

This would explain the impact it has on horizontal use of methodologies. However, only 2 of the 11 elements comprising the deployment of methodologies are related to maturity. The relationship is thus not as strong as expected, but as was shown in section 5.4.2, the effect that CMM level 3 key practices have on the deployment of methodologies is substantial. The lack of significance of Level 2 and 4 key practices with that of deployment supports the finding by Jiang *et al.* (2004), that the benefits of using the CMM may only be realised once Level 3 maturity is attained, and that the benefits may taper off after Level 3. This could possibly be the reason for the weak relationship between the maturity cluster variable and the deployment of systems development methodologies

The time spent on planning, analysis and design is not related to the deployment of systems development methodologies. However, the time spent by an IS department on the development of a new system does influence the deployment of methodologies, albeit only 2 of the 11 elements comprising deployment.

The effect of organisational culture on the deployment of systems development methodologies is shown in Table 7.4. Developmental culture is the most influential on deployment of methodologies. These results were discussed thoroughly in section 6.4.2 and will not be discussed again here.

Table 7.6 compares the results of this study to those of Huisman (1999). It can be seen that IS department size was influential on the deployment of systems development methodologies in 1999, as it is found to be in this study. Maturity has an effect on the support for organisational alignment and horizontal use, whereas it influenced horizontal use and vertical use in Huisman (1999). There was no significant relationship between deployment and the time spent on planning, analysis and design in both the studies.

The time spent on new development has approximately the same effect on the deployment of systems development methodologies in both the studies.

The findings of this chapter indicate that organisational factors such as the size of the IS department, maturity and the time spent on new development do indeed have an effect on the deployment of systems development methodologies just as they did in 1999 when Huisman conducted her research. However, in this study, the organisational factors have a relationship

with different aspects of deployment such as, support as cognitive and co-operative technology, and impact on the development process.

Both of the studies found that no relationship exists between deployment of systems development methodologies and the time taken for the planning, analysis and design of the system.

A significant finding in this study is that developmental culture has a strong relationship to deployment of methodologies which is unlike the result obtained by Huisman (1999), who found hierarchical culture to be most influential on deployment. These two culture groups are complete opposites of each other and it implies that a culture shift has taken place since 1999.

CHAPTER 8

THE INDIVIDUAL DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

8.1 INTRODUCTION

This chapter will discuss the deployment of systems development methodologies amongst individual systems developers. When a decision to use a systems development methodology is made by the management of the IS department, it does not guarantee that all developers will readily accept this decision and use the specified methodology. This chapter presents the findings of an investigation into the factors that influence the individual deployment of systems development methodologies.

The factors that were investigated will be introduced, followed by a short discussion of each after which the elements that comprise the deployment of systems development methodologies are summarised. The results will then be discussed and compared to Huisman (1999) followed by conclusions drawn and final remarks.

8.2 INDIVIDUAL CHARACTERISTICS AND THE DEPLOYMENT OF SYSTEMS DEVELOPMENT METHODOLOGIES

The factors examined in the analysis are summarised in Table 8.1. The characteristics and their corresponding factors were selected according to the conceptual research model which was defined and tested by Huisman (1999) and discussed in detail in section 3.2. The measurement of the factors was done with the instrument of measurement which was tested for reliability and used by Huisman (1999).

Characteristics	Factors
Innovation	Relative Advantage Complexity (Ease of use) Compatibility Demonstrability Triability
Individual	Education Experience in Systems Development Experience with Systems Development Methodologies
Task	Time Spent on the Development of New Applications Time Spent on the Planning, Analysis and Design of Systems
Organisational	Management Support Developer Support
Environmental	Uncertainty
Innovation Diffusion Process	Voluntariness

Table 8.1 Factors Studied to Determine Effect on Individual Deployment of Systems Development Methodologies

8.2.1 Innovation Characteristics

- ***Relative Advantage***

The degree to which an innovation is perceived as being superior to the idea that supplants it, is termed as the relative advantage of the innovation (Rogers, 1995). A new innovation should generally be better than the innovation that it will replace.

- ***Complexity (Ease of use)***

Complexity is the extent to which an innovation is perceived as difficult to understand and use (Rogers, 1995). It is human nature that complex tasks are generally avoided. Thus the more complex an innovation is thought to be, the less likely it is that it will be adopted and implemented.

- ***Compatibility***

The degree of consistency that an innovation has with existing values, past experiences and needs of potential adapters, and positive relation to innovation use, is called the compatibility of an innovation (Rogers, 1995). A new innovation must “fit” into the context of its implementation.

- ***Demonstrability***

The extent to which an innovation's results are visible to others is termed as its demonstrability (Rogers, 1995). When potential adopters of a new innovation actually see the results that it produces, it can positively influence their decision to implement the innovation.

- ***Trialability***

The degree to which an innovation may be tested, or experimented with, is the trialability of the innovation. The more exposure that a new innovation is given, the less uncertainty there is around it, and the faster it will be adopted (Rogers, 1995).

8.2.2 Individual Characteristics

- ***Education***

The level of education of individual developers may possibly have an effect on the deployment of systems development methodologies. This postulation is tested in the analysis.

- ***Experience in Systems Development***

It is speculated that level of experience that a developer has in the systems development field is a determining factor in the adoption of systems development methodologies and CASE tools. This is tested in the analysis.

- ***Experience in Systems Development Methodologies***

Developers that have experience in systems development methodologies might be more willing to adopt them since they have been exposed to the pros and cons of using methodologies and have experienced possible benefits from them.

8.2.3 Task Characteristics

Task characteristics were measured by asking developers to indicate the percentage of effort they dedicated to the development of new applications, and to systems planning, analysis and design.

8.2.4 Organisational Characteristics

- ***Management Support***

The adoption of any technology or innovation is simplified if there is sufficient support from management (Van de Velde, 1992; Senn and Wynekoop, 1995). This relation is narrowed to systems development methodologies and investigated in the analysis that follows.

- ***Developer Support***

The successful adoption of a new innovation can only occur if the individuals that are meant to use it, actually commit to it, and make a concerted effort to implement it (Kwon and Zmud, 1987; Sumner, 1992).

8.2.5 Environmental Characteristics

- ***Uncertainty***

Methodologies are thought to be implemented more commonly in large organisations whose IS departments are not under threat of being disbanded. There is unfortunately no empirical research available on this topic, however, this study attempts to investigate this relationship and draw a conclusion from it.

8.2.6 Innovation Diffusion Process Characteristics

- ***Voluntariness***

The degree to which an innovation is perceived as being adopted voluntarily or of free will is termed as the voluntariness of the innovation (Moore and Benbasat, 1991).

8.2.7 Deployment of Systems Development Methodologies

The deployment of systems development methodologies was studied using the three perspectives described below (Huisman, 1999).

- The use of systems development methodologies
- The perceived support that systems development methodologies provide
- The perceived impact of systems development methodologies

These three perspectives were discussed in detail in section 5.3 and thus only an overview of each is provided now.

• *Use of Systems Development Methodologies*

The use of systems development methodologies is measured by focusing on horizontal and vertical use.

- *Horizontal Use* - Horizontal use relates to the frequency of methodology use by the individual developers.
- *Vertical Use* - Vertical use relates to the intensity of use of systems development methodologies. The maximum of the all the indicated methods was selected for this measure.

• *Perceived Methodology Support*

Methodology support is measured by each of the following components (Huisman, 1999):

- Perceived Support as Production Technology
- Perceived Support as Control Technology
- Perceived Support as Cognitive and Co-operative Technology

- ***Perceived Impact of Systems Development Methodologies***

Two aspects need to be considered in studying the impact of systems development methodologies (Huisman, 1999):

- The impact of the methodology on the quality of the product, i.e. the developed system.
- The impact of the methodology on the quality and productivity of the *process* of systems development.

8.3 RESULTS

Simple linear regression was performed on the data to investigate the effect that the individual characteristics have on the deployment of systems development methodologies. A decision was taken to perform simple regression instead of multiple regression based on the advice received from Statistical Consultation Services. They also recommended that instead of using the *p-value* as a measure of statistical significance, the *Adjusted R²* value be used.

The reasoning behind this was that the *p-value* assumes that a random sample population is used and that the population is large. In contrast, the *Adjusted R²* value is a statistical measure that is used to deliver comment on fixed samples that are not very large. The latter is the case for this study. The *Adjusted R²* value is significant when it is ≥ 0.25 .

The independent variables used in this analysis were obtained by taking the mean of the items in the questionnaire that described each of the factors which comprise the various characteristics for each of the organisations. The different measurements of deployment were used as the dependant variables.

Table 8.2 shows the effect that three of the five innovation characteristics have on the deployment of systems development methodologies. A strong relation exists between each of the three factors and the deployment of methodologies.

Table 8.3 depicts the effect that the remaining two innovation characteristics have on the deployment of methodologies. It also shows the impact that the uncertainty about the future of the IS department has on the deployment of methodologies.

Individual characteristics and their impact on the deployment of systems development methodologies are shown in Table 8.4. There is a very weak relation between the dependant and independent variables in the analysis.

The impact of organisational and innovation diffusion process characteristics on the deployment of systems development methodologies is portrayed in Table 8.5. The support provided by managers has no bearing on the deployment of methodologies. However, the support offered by developers does have an effect on the deployment of systems development methodologies.

Table 8.6 shows that task characteristics have no impact on the deployment of systems development methodologies.

Aspect of Deployment	Relative Advantage	Complexity (Ease of use)	Compatibility
Support as Production Technology	$R^2 = 0.66$ Adjusted $R^2 = 0.63^*$ $F = 19.56$ $p < 0.01$ $\beta = 0.81$	$R^2 = 0.51$ Adjusted $R^2 = 0.46^*$ $F = 10.48$ $p = 0.01$ $\beta = 0.72$	$R^2 = 0.74$ Adjusted $R^2 = 0.72^*$ $F = 29.20$ $p < 0.001$ $\beta = 0.86$
Support for Organisational Alignment	$R^2 = 0.42$ Adjusted $R^2 = 0.37^*$ $F = 7.38$ $p = 0.02$ $\beta = 0.65$	$R^2 = 0.35$ Adjusted $R^2 = 0.29^*$ $F = 5.48$ $p = 0.04$ $\beta = 0.60$	$R^2 = 0.48$ Adjusted $R^2 = 0.42^*$ $F = 9.07$ $p = 0.01$ $\beta = 0.69$
Support for Technical Design	$R^2 = 0.77$ Adjusted $R^2 = 0.75^*$ $F = 33.68$ $p < 0.001$ $\beta = 0.88$	$R^2 = 0.66$ Adjusted $R^2 = 0.63^*$ $F = 19.75$ $p < 0.01$ $\beta = 0.81$	$R^2 = 0.85$ Adjusted $R^2 = 0.84^*$ $F = 57.10$ $p < 0.0001$ $\beta = 0.92$
Support for Verification and Validation	$R^2 = 0.67$ Adjusted $R^2 = 0.64^*$ $F = 20.15$ $p < 0.01$ $\beta = 0.82$	$R^2 = 0.35$ Adjusted $R^2 = 0.29^*$ $F = 5.45$ $p = 0.04$ $\beta = 0.60$	$R^2 = 0.76$ Adjusted $R^2 = 0.74^*$ $F = 32.14$ $p < 0.001$ $\beta = 0.87$
Support as Control Technology	$R^2 = 0.65$ Adjusted $R^2 = 0.61^*$ $F = 18.19$ $p < 0.01$ $\beta = 0.80$	$R^2 = 0.50$ Adjusted $R^2 = 0.45^*$ $F = 10.02$ $p = 0.01$ $\beta = 0.71$	$R^2 = 0.71$ Adjusted $R^2 = 0.68^*$ $F = 24.79$ $p < 0.001$ $\beta = 0.84$
Support as Cognitive and Co-operative Technology	$R^2 = 0.62$ Adjusted $R^2 = 0.58^*$ $F = 14.96$ $p < 0.01$ $\beta = 0.79$	$R^2 = 0.67$ Adjusted $R^2 = 0.63^*$ $F = 18.39$ $p < 0.01$ $\beta = 0.82$	$R^2 = 0.67$ Adjusted $R^2 = 0.63^*$ $F = 17.97$ $p < 0.01$ $\beta = 0.82$
Support for Common Conception of SD Practice	$R^2 = 0.71$ Adjusted $R^2 = 0.67^*$ $F = 21.70$ $p < 0.01$ $\beta = 0.84$	$R^2 = 0.64$ Adjusted $R^2 = 0.60^*$ $F = 15.69$ $p < 0.01$ $\beta = 0.80$	$R^2 = 0.76$ Adjusted $R^2 = 0.73^*$ $F = 28.32$ $p < 0.001$ $\beta = 0.87$
Support for the Evaluation of SD Practice	$R^2 = 0.03$ Adjusted $R^2 = -0.08$ $F = 0.27$ $p = 0.62$ $\beta = 0.17$	$R^2 = 0.30$ Adjusted $R^2 = 0.23$ $F = 3.91$ $p = 0.08$ $\beta = 0.55$	$R^2 = 0.03$ Adjusted $R^2 = -0.08$ $F = 0.27$ $p = 0.62$ $\beta = 0.17$
Impact on the Developed System	$R^2 = 0.71$ Adjusted $R^2 = 0.68^*$ $F = 22.15$ $p < 0.01$ $\beta = 0.84$	$R^2 = 0.63$ Adjusted $R^2 = 0.58^*$ $F = 15.08$ $p < 0.01$ $\beta = 0.80$	$R^2 = 0.76$ Adjusted $R^2 = 0.74^*$ $F = 29.02$ $p < 0.001$ $\beta = 0.87$
Impact on the Development Process	$R^2 = 0.75$ Adjusted $R^2 = 0.72^*$ $F = 26.43$ $p < 0.01$ $\beta = 0.86$	$R^2 = 0.69$ Adjusted $R^2 = 0.65^*$ $F = 19.93$ $p < 0.01$ $\beta = 0.83$	$R^2 = 0.80$ Adjusted $R^2 = 0.78^*$ $F = 35.88$ $p < 0.001$ $\beta = 0.89$
Productivity Effects and Morale	$R^2 = 0.65$ Adjusted $R^2 = 0.61^*$ $F = 16.59$ $p < 0.01$ $\beta = 0.81$	$R^2 = 0.65$ Adjusted $R^2 = 0.61^*$ $F = 16.56$ $p < 0.01$ $\beta = 0.80$	$R^2 = 0.83$ Adjusted $R^2 = 0.65^*$ $F = 19.44$ $p < 0.01$ $\beta = 0.83$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.73$ Adjusted $R^2 = 0.70^*$ $F = 24.05$ $p < 0.001$ $\beta = 0.85$	$R^2 = 0.63$ Adjusted $R^2 = 0.59^*$ $F = 15.16$ $p < 0.01$ $\beta = 0.80$	$R^2 = 0.79$ Adjusted $R^2 = 0.78^*$ $F = 33.95$ $p < 0.001$ $\beta = 0.89$
Horizontal Use	$R^2 = 0.14$ Adjusted $R^2 = 0.07$ $F = 2.09$ $p = 0.17$ $\beta = 0.37$	$R^2 = 0.16$ Adjusted $R^2 = 0.09$ $F = 2.46$ $p = 0.14$ $\beta = 0.40$	$R^2 = 0.02$ Adjusted $R^2 = -0.05$ $F = 0.27$ $p = 0.61$ $\beta = 0.14$
Vertical Use	$R^2 = 0.00$ Adjusted $R^2 = -0.08$ $F = 0.00$ $p = 0.99$ $\beta = -0.00$	$R^2 = 0.08$ Adjusted $R^2 = 0.01$ $F = 1.10$ $p = 0.31$ $\beta = 0.29$	$R^2 = 0.03$ Adjusted $R^2 = -0.05$ $F = 0.38$ $p = 0.55$ $\beta = -0.18$

Table 8.2 The Impact of Relative Advantage, Complexity and Compatibility on the Deployment of Systems Development Methodologies

Aspect of Deployment	Demonstrability	Trialability	Uncertainty
Support as Production Technology	$R^2 = 0.38$ Adjusted $R^2 = 0.32^*$ $F = 6.12$ $p = 0.03$ $\beta = 0.62$	$R^2 = 0.55$ Adjusted $R^2 = 0.50^*$ $F = 12.20$ $p = 0.01$ $\beta = 0.74$	$R^2 = 0.12$ Adjusted $R^2 = 0.03^*$ $F = 1.34$ $p = 0.27$ $\beta = -0.34$
Support for Organisational Alignment	$R^2 = 0.40$ Adjusted $R^2 = 0.34^*$ $F = 6.64$ $p = 0.03$ $\beta = 0.63$	$R^2 = 0.37$ Adjusted $R^2 = 0.30^*$ $F = 5.82$ $p = 0.04$ $\beta = 0.61$	$R^2 = 0.15$ Adjusted $R^2 = 0.07$ $F = 1.82$ $p = 0.21$ $\beta = -0.39$
Support for Technical Design	$R^2 = 0.35$ Adjusted $R^2 = 0.28^*$ $F = 5.45$ $p = 0.04$ $\beta = 0.59$	$R^2 = 0.61$ Adjusted $R^2 = 0.57^*$ $F = 15.86$ $p < 0.01$ $\beta = 0.78$	$R^2 = 0.09$ Adjusted $R^2 = 0.00$ $F = 1.03$ $p = 0.33$ $\beta = -0.31$
Support for Verification and Validation	$R^2 = 0.26$ Adjusted $R^2 = 0.18$ $F = 3.50$ $p = 0.09$ $\beta = 0.51$	$R^2 = 0.59$ Adjusted $R^2 = 0.55^*$ $F = 14.29$ $p < 0.01$ $\beta = 0.77$	$R^2 = 0.04$ Adjusted $R^2 = -0.05$ $F = 0.47$ $p = 0.51$ $\beta = -0.21$
Support as Control Technology	$R^2 = 0.28$ Adjusted $R^2 = 0.21$ $F = 3.92$ $p = 0.08$ $\beta = 0.53$	$R^2 = 0.67$ Adjusted $R^2 = 0.64^*$ $F = 20.21$ $p < 0.01$ $\beta = 0.82$	$R^2 = 0.13$ Adjusted $R^2 = 0.04$ $F = 1.50$ $p = 0.25$ $\beta = -0.36$
Support as Cognitive and Co-operative Technology	$R^2 = 0.30$ Adjusted $R^2 = 0.23$ $F = 3.94$ $p = 0.08$ $\beta = 0.55$	$R^2 = 0.79$ Adjusted $R^2 = 0.77^*$ $F = 34.41$ $p < 0.001$ $\beta = 0.89$	$R^2 = 0.02$ Adjusted $R^2 = -0.08$ $F = 0.22$ $p = 0.65$ $\beta = -0.15$
Support for Common Conception of SD Practice	$R^2 = 0.24$ Adjusted $R^2 = 0.16$ $F = 2.90$ $p = 0.12$ $\beta = 0.49$	$R^2 = 0.76$ Adjusted $R^2 = 0.74^*$ $F = 29.24$ $p < 0.001$ $\beta = 0.87$	$R^2 = 0.06$ Adjusted $R^2 = -0.04$ $F = 0.62$ $p = 0.45$ $\beta = -0.25$
Support for the Evaluation of SD Practice	$R^2 = 0.31$ Adjusted $R^2 = 0.23$ $F = 3.99$ $p = 0.08$ $\beta = 0.55$	$R^2 = 0.32$ Adjusted $R^2 = 0.24$ $F = 4.19$ $p = 0.07$ $\beta = 0.56$	$R^2 = 0.11$ Adjusted $R^2 = 0.01$ $F = 1.08$ $p = 0.33$ $\beta = 0.33$
Impact on the Developed System	$R^2 = 0.20$ Adjusted $R^2 = 0.11$ $F = 2.24$ $p = 0.17$ $\beta = 0.45$	$R^2 = 0.65$ Adjusted $R^2 = 0.62^*$ $F = 17.02$ $p < 0.01$ $\beta = 0.81$	$R^2 = 0.30$ Adjusted $R^2 = 0.22$ $F = 3.82$ $p = 0.08$ $\beta = -0.55$
Impact on the Development Process	$R^2 = 0.19$ Adjusted $R^2 = 0.10$ $F = 2.13$ $p = 0.18$ $\beta = 0.44$	$R^2 = 0.65$ Adjusted $R^2 = 0.61^*$ $F = 16.58$ $p < 0.01$ $\beta = 0.81$	$R^2 = 0.17$ Adjusted $R^2 = 0.08$ $F = 1.84$ $p = 0.21$ $\beta = -0.41$
Productivity Effects and Morale	$R^2 = 0.12$ Adjusted $R^2 = 0.02$ $F = 1.24$ $p = 0.29$ $\beta = 0.35$	$R^2 = 0.53$ Adjusted $R^2 = 0.48^*$ $F = 10.16$ $p = 0.01$ $\beta = 0.73$	$R^2 = 0.39$ Adjusted $R^2 = 0.32^*$ $F = 5.77$ $p = 0.04$ $\beta = -0.63$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.24$ Adjusted $R^2 = 0.15$ $F = 2.77$ $p = 0.13$ $\beta = 0.49$	$R^2 = 0.66$ Adjusted $R^2 = 0.63^*$ $F = 17.79$ $p < 0.01$ $\beta = 0.81$	$R^2 = 0.04$ Adjusted $R^2 = -0.07$ $F = 0.35$ $p = 0.57$ $\beta = -0.19$
Horizontal Use	$R^2 = 0.55$ Adjusted $R^2 = 0.52^*$ $F = 15.92$ $p < 0.01$ $\beta = 0.74$	$R^2 = 0.11$ Adjusted $R^2 = 0.04$ $F = 1.54$ $p = 0.24$ $\beta = 0.33$	$R^2 = 0.03$ Adjusted $R^2 = -0.05$ $F = 0.36$ $p = 0.56$ $\beta = -0.17$
Vertical Use	$R^2 = 0.25$ Adjusted $R^2 = 0.19$ $F = 4.03$ $p = 0.07$ $\beta = 0.50$	$R^2 = 0.01$ Adjusted $R^2 = -0.07$ $F = 0.17$ $p = 0.69$ $\beta = 0.12$	$R^2 = 0.00$ Adjusted $R^2 = -0.08$ $F = 0.03$ $p = 0.87$ $\beta = 0.05$

Table 8.3 The Impact of Demonstrability, Trialability and Environmental Characteristics on the Deployment of Systems Development Methodologies

Aspect of Deployment	Education	Experience in Systems Development	Systems Development Methodology Experience
Support as Production Technology	$R^2 = 0.02$ Adjusted $R^2 = -0.08$ $F = 0.20$ $p = 0.67$ $\beta = -0.14$	$R^2 = 0.14$ Adjusted $R^2 = 0.06$ $F = 1.64$ $p = 0.23$ $\beta = -0.38$	$R^2 = 0.06$ Adjusted $R^2 = -0.03$ $F = 0.68$ $p = 0.43$ $\beta = 0.25$
Support for Organisational Alignment	$R^2 = 0.02$ Adjusted $R^2 = -0.08$ $F = 0.15$ $p = 0.70$ $\beta = -0.12$	$R^2 = 0.15$ Adjusted $R^2 = 0.06$ $F = 1.70$ $p = 0.22$ $\beta = -0.38$	$R^2 = 0.02$ Adjusted $R^2 = -0.08$ $F = 0.18$ $p = 0.68$ $\beta = 0.13$
Support for Technical Design	$R^2 = 0.03$ Adjusted $R^2 = -0.07$ $F = 0.26$ $p = 0.62$ $\beta = -0.16$	$R^2 = 0.06$ Adjusted $R^2 = -0.04$ $F = 0.59$ $p = 0.46$ $\beta = -0.24$	$R^2 = 0.18$ Adjusted $R^2 = 0.10$ $F = 2.22$ $p = 0.17$ $\beta = 0.43$
Support for Verification and Validation	$R^2 = 0.00$ Adjusted $R^2 = -0.10$ $F = 0.02$ $p = 0.89$ $\beta = -0.05$	$R^2 = 0.32$ Adjusted $R^2 = 0.25^*$ $F = 4.63$ $p = 0.06$ $\beta = -0.56$	$R^2 = 0.01$ Adjusted $R^2 = -0.09$ $F = 0.13$ $p = 0.73$ $\beta = 0.11$
Support as Control Technology	$R^2 = 0.02$ Adjusted $R^2 = -0.07$ $F = 0.23$ $p = 0.64$ $\beta = -0.14$	$R^2 = 0.18$ Adjusted $R^2 = 0.11$ $F = 2.43$ $p = 0.15$ $\beta = -0.43$	$R^2 = 0.00$ Adjusted $R^2 = -0.09$ $F = 0.02$ $p = 0.88$ $\beta = 0.05$
Support as Cognitive and Co-operative Technology	$R^2 = 0.04$ Adjusted $R^2 = -0.07$ $F = 0.34$ $p = 0.58$ $\beta = -0.19$	$R^2 = 0.17$ Adjusted $R^2 = 0.08$ $F = 1.89$ $p = 0.20$ $\beta = -0.42$	$R^2 = 0.17$ Adjusted $R^2 = 0.08$ $F = 1.90$ $p = 0.20$ $\beta = 0.42$
Support for Common Conception of SD Practice	$R^2 = 0.01$ Adjusted $R^2 = -0.10$ $F = 0.10$ $p = 0.76$ $\beta = -0.10$	$R^2 = 0.19$ Adjusted $R^2 = 0.09$ $F = 2.05$ $p = 0.19$ $\beta = -0.43$	$R^2 = 0.15$ Adjusted $R^2 = 0.06$ $F = 1.62$ $p = 0.24$ $\beta = 0.39$
Support for the Evaluation of SD Practice	$R^2 = 0.24$ Adjusted $R^2 = 0.16$ $F = 2.83$ $p = 0.13$ $\beta = -0.49$	$R^2 = 0.04$ Adjusted $R^2 = -0.07$ $F = 0.37$ $p = 0.56$ $\beta = -0.20$	$R^2 = 0.09$ Adjusted $R^2 = -0.01$ $F = 0.91$ $p = 0.36$ $\beta = 0.30$
Impact on the Developed System	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.00$ $p = 0.98$ $\beta = -0.01$	$R^2 = 0.21$ Adjusted $R^2 = 0.12$ $F = 2.32$ $p = 0.16$ $\beta = -0.45$	$R^2 = 0.07$ Adjusted $R^2 = -0.04$ $F = 0.64$ $p = 0.44$ $\beta = 0.26$
Impact on the Development Process	$R^2 = 0.03$ Adjusted $R^2 = -0.07$ $F = 0.32$ $p = 0.59$ $\beta = -0.18$	$R^2 = 0.07$ Adjusted $R^2 = -0.03$ $F = 0.72$ $p = 0.42$ $\beta = -0.27$	$R^2 = 0.22$ Adjusted $R^2 = 0.13$ $F = 2.47$ $p = 0.15$ $\beta = 0.46$
Productivity Effects and Morale	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.01$ $p = 0.94$ $\beta = -0.03$	$R^2 = 0.07$ Adjusted $R^2 = -0.04$ $F = 0.65$ $p = 0.44$ $\beta = -0.26$	$R^2 = 0.15$ Adjusted $R^2 = 0.06$ $F = 1.59$ $p = 0.24$ $\beta = 0.39$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.10$ Adjusted $R^2 = -0.00$ $F = 0.96$ $p = 0.35$ $\beta = -0.31$	$R^2 = 0.07$ Adjusted $R^2 = -0.03$ $F = 0.68$ $p = 0.43$ $\beta = -0.27$	$R^2 = 0.25$ Adjusted $R^2 = 0.16$ $F = 2.97$ $p = 0.12$ $\beta = 0.50$
Horizontal Use	$R^2 = 0.26$ Adjusted $R^2 = 0.21$ $F = 5.02$ $p = 0.04$ $\beta = 0.51$	$R^2 = 0.01$ Adjusted $R^2 = -0.06$ $F = 0.15$ $p = 0.70$ $\beta = -0.10$	$R^2 = 0.20$ Adjusted $R^2 = 0.14$ $F = 3.47$ $p = 0.08$ $\beta = 0.45$
Vertical Use	$R^2 = 0.03$ Adjusted $R^2 = -0.06$ $F = 0.31$ $p = 0.59$ $\beta = 0.16$	$R^2 = 0.08$ Adjusted $R^2 = -0.00$ $F = 0.99$ $p = 0.34$ $\beta = 0.28$	$R^2 = 0.06$ Adjusted $R^2 = -0.02$ $F = 0.76$ $p = 0.40$ $\beta = 0.24$

Table 8.4 The Impact of Individual Characteristics on the Deployment of Systems Development Methodologies

Aspect of Deployment	Manager Support	Developer Support	Voluntariness
Support as Production Technology	$R^2 = 0.13$ Adjusted $R^2 = 0.04$ $F = 1.46$ $p = 0.26$ $\beta = 0.36$	$R^2 = 0.22$ Adjusted $R^2 = 0.15$ $F = 2.87$ $p = 0.12$ $\beta = 0.47$	$R^2 = 0.63$ Adjusted $R^2 = 0.60^*$ $F = 17.21$ $p < 0.01$ $\beta = -0.80$
Support for Organisational Alignment	$R^2 = 0.11$ Adjusted $R^2 = 0.02$ $F = 1.26$ $p = 0.29$ $\beta = 0.33$	$R^2 = 0.11$ Adjusted $R^2 = 0.02$ $F = 1.26$ $p = 0.29$ $\beta = 0.33$	$R^2 = 0.74$ Adjusted $R^2 = 0.71^*$ $F = 27.76$ $p < 0.001$ $\beta = -0.86$
Support for Technical Design	$R^2 = 0.14$ Adjusted $R^2 = 0.05$ $F = 1.64$ $p = 0.23$ $\beta = 0.38$	$R^2 = 0.37$ Adjusted $R^2 = 0.31^*$ $F = 5.98$ $p = 0.03$ $\beta = 0.61$	$R^2 = 0.54$ Adjusted $R^2 = 0.50^*$ $F = 11.97$ $p = 0.01$ $\beta = -0.74$
Support for Verification and Validation	$R^2 = 0.09$ Adjusted $R^2 = -0.00$ $F = 0.95$ $p = 0.35$ $\beta = 0.29$	$R^2 = 0.11$ Adjusted $R^2 = 0.02$ $F = 1.26$ $p = 0.29$ $\beta = 0.33$	$R^2 = 0.35$ Adjusted $R^2 = 0.28^*$ $F = 5.35$ $p = 0.04$ $\beta = -0.59$
Support as Control Technology	$R^2 = 0.15$ Adjusted $R^2 = 0.07$ $F = 1.79$ $p = 0.21$ $\beta = 0.39$	$R^2 = 0.18$ Adjusted $R^2 = 0.10$ $F = 2.26$ $p = 0.16$ $\beta = 0.43$	$R^2 = 0.28$ Adjusted $R^2 = 0.22$ $F = 4.31$ $p = 0.06$ $\beta = -0.53$
Support as Cognitive and Co-operative Technology	$R^2 = 0.15$ Adjusted $R^2 = 0.06$ $F = 1.60$ $p = 0.24$ $\beta = 0.39$	$R^2 = 0.36$ Adjusted $R^2 = 0.29^*$ $F = 5.11$ $p = 0.05$ $\beta = 0.60$	$R^2 = 0.22$ Adjusted $R^2 = 0.13$ $F = 2.55$ $p = 0.14$ $\beta = -0.47$
Support for Common Conception of SD Practice	$R^2 = 0.19$ Adjusted $R^2 = 0.10$ $F = 2.11$ $p = 0.18$ $\beta = 0.44$	$R^2 = 0.40$ Adjusted $R^2 = 0.33^*$ $F = 5.95$ $p = 0.04$ $\beta = 0.63$	$R^2 = 0.20$ Adjusted $R^2 = 0.11$ $F = 2.27$ $p = 0.17$ $\beta = -0.45$
Support for the Evaluation of SD Practice	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.00$ $p = 0.99$ $\beta = -0.01$	$R^2 = 0.04$ Adjusted $R^2 = -0.07$ $F = 0.37$ $p = 0.56$ $\beta = 0.20$	$R^2 = 0.11$ Adjusted $R^2 = 0.01$ $F = 1.10$ $p = 0.32$ $\beta = -0.33$
Impact on the Developed System	$R^2 = 0.14$ Adjusted $R^2 = 0.04$ $F = 1.46$ $p = 0.26$ $\beta = 0.37$	$R^2 = 0.40$ Adjusted $R^2 = 0.34^*$ $F = 6.11$ $p = 0.04$ $\beta = 0.64$	$R^2 = 0.36$ Adjusted $R^2 = 0.23$ $F = 4.01$ $p = 0.08$ $\beta = -0.55$
Impact on the Development Process	$R^2 = 0.17$ Adjusted $R^2 = 0.08$ $F = 1.85$ $p = 0.21$ $\beta = 0.41$	$R^2 = 0.55$ Adjusted $R^2 = 0.50^*$ $F = 10.99$ $p = 0.01$ $\beta = 0.74$	$R^2 = 0.31$ Adjusted $R^2 = 0.33^*$ $F = 5.95$ $p = 0.04$ $\beta = 0.63$
Productivity Effects and Morale	$R^2 = 0.18$ Adjusted $R^2 = 0.10$ $F = 2.04$ $p = 0.19$ $\beta = 0.43$	$R^2 = 0.56$ Adjusted $R^2 = 0.51^*$ $F = 11.31$ $p = 0.01$ $\beta = 0.75$	$R^2 = 0.25$ Adjusted $R^2 = 0.16$ $F = 2.96$ $p = 0.12$ $\beta = -0.50$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.14$ Adjusted $R^2 = 0.04$ $F = 1.42$ $p = 0.26$ $\beta = 0.37$	$R^2 = 0.47$ Adjusted $R^2 = 0.41^*$ $F = 7.89$ $p = 0.02$ $\beta = 0.68$	$R^2 = 0.32$ Adjusted $R^2 = 0.24$ $F = 4.24$ $p = 0.07$ $\beta = -0.57$
Horizontal Use	$R^2 = 0.26$ Adjusted $R^2 = 0.20$ $F = 4.5$ $p = 0.05$ $\beta = 0.51$	$R^2 = 0.11$ Adjusted $R^2 = 0.04$ $F = 1.57$ $p = 0.23$ $\beta = 0.33$	$R^2 = 0.45$ Adjusted $R^2 = 0.41^*$ $F = 11.33$ $p < 0.01$ $\beta = -0.67$
Vertical Use	$R^2 = 0.14$ Adjusted $R^2 = 0.07$ $F = 1.91$ $p = 0.19$ $\beta = 0.37$	$R^2 = 0.07$ Adjusted $R^2 = -0.01$ $F = 0.86$ $p = 0.37$ $\beta = 0.26$	$R^2 = 0.11$ Adjusted $R^2 = 0.04$ $F = 1.55$ $p = 0.24$ $\beta = -0.34$

Table 8.5 The Impact of Organisational and Innovation Diffusion Process Characteristics on the Deployment of Systems Development Methodologies

Aspect of Deployment	Time: Planning, Analysis, Design	Time: New Development
Support as Production Technology	$R^2 = 0.00$ Adjusted $R^2 = -0.10$ $F = 0.00$ $p = 0.98$ $\beta = 0.01$	$R^2 = 0.07$ Adjusted $R^2 = -0.02$ $F = 0.80$ $p = 0.39$ $\beta = 0.27$
Support for Organisational Alignment	$R^2 = 0.03$ Adjusted $R^2 = -0.06$ $F = 0.37$ $p = 0.56$ $\beta = 0.19$	$R^2 = 0.15$ Adjusted $R^2 = 0.07$ $F = 1.80$ $p = 0.21$ $\beta = 0.39$
Support for Technical Design	$R^2 = 0.02$ Adjusted $R^2 = -0.08$ $F = 0.16$ $p = 0.70$ $\beta = -0.13$	$R^2 = 0.01$ Adjusted $R^2 = -0.09$ $F = 0.13$ $p = 0.73$ $\beta = 0.11$
Support for Verification and Validation	$R^2 = 0.00$ Adjusted $R^2 = -0.10$ $F = 0.05$ $p = 0.83$ $\beta = -0.07$	$R^2 = 0.07$ Adjusted $R^2 = -0.02$ $F = 0.74$ $p = 0.41$ $\beta = 0.26$
Support as Control Technology	$R^2 = 0.00$ Adjusted $R^2 = -0.08$ $F = 0.07$ $p = 0.80$ $\beta = -0.08$	$R^2 = 0.00$ Adjusted $R^2 = -0.09$ $F = 0.03$ $p = 0.87$ $\beta = 0.05$
Support as Cognitive and Co-operative Technology	$R^2 = 0.18$ Adjusted $R^2 = 0.09$ $F = 2.02$ $p = 0.19$ $\beta = -0.43$	$R^2 = 0.02$ Adjusted $R^2 = -0.09$ $F = 0.20$ $p = 0.66$ $\beta = -0.15$
Support for Common Conception of SD Practice	$R^2 = 0.22$ Adjusted $R^2 = -0.14$ $F = 2.6$ $p = 0.14$ $\beta = -0.47$	$R^2 = 0.03$ Adjusted $R^2 = -0.08$ $F = 0.24$ $p = 0.64$ $\beta = -0.16$
Support for the Evaluation of SD Practice	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.02$ $p = 0.90$ $\beta = -0.04$	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.02$ $p = 0.90$ $\beta = -0.04$
Impact on the Developed System	$R^2 = 0.10$ Adjusted $R^2 = -0.00$ $F = 0.96$ $p = 0.35$ $\beta = -0.31$	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.01$ $p = 0.92$ $\beta = -0.03$
Impact on the Development Process	$R^2 = 0.20$ Adjusted $R^2 = 0.11$ $F = 2.31$ $p = 0.16$ $\beta = -0.45$	$R^2 = 0.02$ Adjusted $R^2 = -0.09$ $F = 0.21$ $p = 0.56$ $\beta = 0.15$
Productivity Effects and Morale	$R^2 = 0.24$ Adjusted $R^2 = 0.16$ $F = 2.92$ $p = 0.12$ $\beta = -0.49$	$R^2 = 0.07$ Adjusted $R^2 = -0.04$ $F = 0.63$ $p = 0.45$ $\beta = -0.26$
Quality Effects, Goal Achievement and Reputation	$R^2 = 0.14$ Adjusted $R^2 = 0.05$ $F = 1.52$ $p = 0.25$ $\beta = -0.38$	$R^2 = 0.00$ Adjusted $R^2 = -0.11$ $F = 0.02$ $p = 0.89$ $\beta = -0.05$
Horizontal Use	$R^2 = 0.00$ Adjusted $R^2 = -0.07$ $F = 0.01$ $p = 0.93$ $\beta = -0.02$	$R^2 = 0.00$ Adjusted $R^2 = -0.07$ $F = 0.01$ $p = 0.93$ $\beta = -0.02$
Vertical Use	$R^2 = 0.04$ Adjusted $R^2 = -0.04$ $F = 0.49$ $p = 0.50$ $\beta = -0.20$	$R^2 = 0.10$ Adjusted $R^2 = 0.03$ $F = 1.35$ $p = 0.27$ $\beta = -0.32$

Table 8.6 The Impact of Task Characteristics on the Deployment of Systems Development Methodologies

Tables 8.7, 8.8 and 8.9 compare the results obtained in this study to those obtained by Huisman (1999). Unlike in this study, Huisman performed factor analysis on the variables. The multiple regression performed by her was thus on the actual variables and not their underlying factors. This means that the independent variables were not tested against the individual elements such as support for technical design, support for organisational alignment, and support for verification and validation which constitute support as production technology. Instead, each of the independent variables was tested against support as production technology as a whole.

In addition to the regression performed on the individual elements of each variable that describe deployment, regression was also performed on the actual variable to facilitate the accurate comparison of results between this study and Huisman (1999). This is reflected in the three tables. The results obtained in this study are represented by a “K” and those obtained by Huisman (1999) are represented by an “H.”

Aspect of Deployment	Relative Advantage	Complexity (ease of use)	Compatibility	Demonstrability	Triability
Support as Production Technology	K, H	K, H	K	K	K
Support for Organisational Alignment	K	K	K	K	K
Support for Technical Design	K	K	K	K	K
Support for Verification and Validation	K	K	K		K
Support as Control Technology	K, H	K	K		K
Support as Cognitive and Co-operative Technology	K, H	K	K		K, H
Support for Common Conception of SD Practice	K	K	K		K
Support for the Evaluation of SD Practice					
Impact on the Developed System	K, H	K	K, H	H	K
Impact on the Development Process	K, H	K	K, H		K
Productivity Effects and Morale	K	K	K		K
Quality Effects, Goal Achievement and Reputation	K	K	K		K
Horizontal Use	H			K	
Vertical Use	H			H	

Table 8.7 Comparison of the Impact of Innovation Characteristics on the Deployment of Systems Development Methodologies between this study and Huisman and Iivari (2002)

Aspect of Deployment	Education	Experience in Systems Development	Experience in Systems Development Methodologies	Manager Support	Developer Support
Support as Production Technology		H		H	H
Support for Organisational Alignment					
Support for Technical Design					K
Support for Verification and Validation		K			
Support as Control Technology				H	
Support as Cognitive and Co-operative Technology					K, H
Support for Common Conception of SD Practice					K
Support for the Evaluation of SD Practice					
Impact on the Developed System					K
Impact on the Development Process					K
Productivity Effects and Morale					K
Quality Effects, Goal Achievement and Reputation					K
Horizontal Use		H	H	H	
Vertical Use			H		

Table 8.8 Comparison of the Impact of Individual and Organisational Characteristics on the Deployment of Systems Development Methodologies between this study and Huisman and Iivari (2002)

Aspect of Deployment	Time: Planning, Analysis, Design	Time: New Development	Uncertainty	Voluntariness
Support as Production Technology			H	K
Support for Organisational Alignment				K
Support for Technical Design				K
Support for Verification and Validation				K
Support as Control Technology			H	
Support as Cognitive and Co-operative Technology		H		
Support for Common Conception of SD Practice				
Support for the Evaluation of SD Practice				
Impact on the Developed System			H	K
Impact on the Development Process			H	
Productivity Effects and Morale			K	
Quality Effects, Goal Achievement and Reputation				
Horizontal Use	H			K
Vertical Use				

Table 8.9 Comparison of the Impact of Task, Environmental and Innovation Diffusion Process Characteristics on the Deployment of Systems Development Methodologies between this study and Huisman and livari (2002)

8.4 DISCUSSION AND FINAL COMMENTS

The results in Table 8.2 and Table 8.3 indicate that a very strong relationship exists between the innovation characteristics comprising of relative advantage, complexity (ease of use), compatibility, demonstrability and trialability, and the deployment of systems development methodologies. This strong relationship is expected since, a new innovation should typically be superior to that which it replaces, it ought to be easy to understand and use, it should be compatible with the existing processes, it must show visible results to the adopters and it should be available for viability testing. The findings prove that the abovementioned is applicable in this study.

The results obtained are very similar to those obtained by Huisman (1999) who also found a relationship between the deployment of systems development methodologies and the innovation characteristics such as ease of use and relative advantage as shown in Table 8.7. The impact that compatibility and trialability have on deployment of systems development methodologies was found to be much stronger than in Huisman (1999).

Five of the seven factors which comprise deployment, i.e. support as production technology, support as control technology, support as cognitive and cooperative technology, impact on the developed system and impact on the development process, have a strong relationship to the two abovementioned innovation characteristics.

Table 8.3 shows the effect that an uncertain IS department future has on the deployment of systems development methodologies. In this study it has a negative effect on the productivity and morale of the individual developers. This is expected, since individuals that are employed in an IS department that has a high risk of being closed, will have a low morale and productivity. Table 8.9 shows that Huisman (1999) found that uncertainty affected the deployment of systems development methodologies to a greater extent than was found in this study. This could possibly be due to the fact the developers in this survey felt more confident and secure about the status of their IS department than those surveyed by Huisman (1999).

The impact that the individual characteristics such as education, experience in systems development and experience in systems development methodologies have on the deployment of methodologies is portrayed in Table 8.4. It can be seen that experience in systems development only has an effect on the support provided for verification and validation. The developers' experience in systems development methodologies has no bearing whatsoever on the deployment of systems development methodologies. The results that Huisman (1999) obtained are shown in Table 8.8. They are similar to the results obtained in this study, apart from the fact that Huisman (1999) found that the experience that a developer has in systems development methodologies has an impact on the use of methodologies, whereas this study revealed no such relationship.

Table 8.5 reveals that the support provided by managers for a new innovation, or a systems development methodology in this case, has no effect on the deployment of that methodology. Developer support however does have an impact on the deployment of methodologies. The elements affected include support for technical design, support for common conception of systems development practice, impact on the developed system, as well as impact on the development process. It is evident that there is quite a strong relation between the variables of methodology deployment and developer support. Huisman (1999) did not find such a strong relation between deployment of methodologies and developer support, however it was found that manager support impacts deployment of methodologies. This is indicated in Table 8.8.

The impact that voluntariness has on the deployment of systems development methodologies is shown in Table 8.5. It was found to have a strong negative effect on 5 of the 11 factors that measure deployment. These factors included, support for organisational deployment, support for technical design, support for verification and validation, impact on the developed system, and horizontal use. Huisman (1999) found that no relation exists between voluntariness and the deployment of systems development methodologies and this is depicted in Table 8.9.

From Table 8.6 it is evident that there is no relation between the task characteristics and the deployment of systems development methodologies. Huisman (1999) found a weak relation to exist between deployment of methodologies and the task characteristics.

It is significant that developer support was found to impact the individual deployment of systems development methodologies whereas manager support had no influence on it. This is opposite to the results of Huisman (1999). Voluntariness was also found to have a significant negative impact on the deployment of methodologies. This could possibly indicate that if given the choice to use a methodology, developers would choose not to and instead have to be coerced into using one.

CHAPTER 9

SUMMARY AND FINAL CONCLUSIONS

9.1 INTRODUCTION

This chapter will summarise the findings of this study and elaborate on its limitations and possible future research into the use and effectiveness of systems development methodologies.

9.2 CONTRIBUTIONS OF THE STUDY

There is a deficiency of empirical research into the use and effectiveness of systems development methodologies. This study contributes to the existing empirical research in this field. Although the study was focused on South Africa, the deployment of systems development was studied at a more abstract level and this makes the findings also significant elsewhere.

The general purpose of this study was to investigate whether there has been a change in the use of SDMs in South Africa since (Huisman, 1999) and also whether the factors that influence the use and effectiveness of SDMs have changed or remained the same as put forth in 1999. This was achieved by attempting to answer the following research questions:

- Has the deployment of SDMs in South Africa changed since 1999?
- Has a change occurred in the relationship between the maturity of an IS department and the deployment of SDMs since 1999?
- Has a change occurred in the relationship between organisational culture and the deployment of SDMs since 1999?
- Has a change occurred in the factors that explain the organisational deployment of SDMs since 1999?
- Has a change occurred in the factors that explain the individual deployment of SDMs since 1999?

Systems development methodologies were defined as the totality of systems development approaches, process models, methods and techniques used in an organisation (Huisman, 1999).

The term “deployment” was used to encompass the following aspects:

- The horizontal and vertical use of systems development methodologies.
- The support provided by the systems development methodology as production, control and cognitive/cooperation technology.
- The impact of systems development methodologies on the developed system and the development process.

9.2.1 The Current State of Systems Development Methodology Use in South Africa

The development environment in South Africa is centred on microcomputers that run mainly Windows 98, UNIX, or Windows XP using a local area network. A wide variety of programming languages are used for the development of new software and the maintenance of legacy systems. The most popular programming languages used are Visual Basic, MSSQL and Java.

Over half of the activities in the departments are devoted to the maintenance of existing systems and approximately 34% percent of the activities are allotted to the development of new applications. This result is similar to that which was obtained by Huisman (1999). Only 6% of the various IS departments’ activities are allocated to package customisation. This contrasts with Huisman (1999), who found that almost 15% of IS departments customise packages. This difference could possibly be explained by the fact that all the organisations that outsourced their software development or purchased packages were not considered in this study.

IS departments spend a third of their efforts on systems planning, analysis and design. Another third of their time is spent on programming, and testing, installation and evaluation are also allocated approximately a third of their time.

The phased process model is the most used, followed by the concurrent, evolutionary, prototyping and spiral models. Although the use of each model is almost evenly distributed, the extent to which each is used should be taken note of. Evolutionary and prototyping process models had above-average use with mean values >3.

The phased process model is reported as having a lower use as compared to 1999 and the use of the spiral model has also declined. Structured process models seem to have declined in popularity, and process models that allow multiple tasks to occur simultaneously, are used more frequently within organisations. The prevalence of prototyping shows that clients want tangible evidence of the finished product before development is completed.

An important finding is that the intensity of use of techniques and methods is very low compared to (Huisman, 1999). Although most organisations use some sort of technique or method, the degree of use is nominal. The low intensity of use supports the theory by Avison and Fitzgerald (2003) that the software development industry is undergoing a paradigm shift and moving into a post-methodology era where organisations are not using methodologies to develop software anymore.

It also supports another theory that Fitzgerald *et al.* (2002) call Methods-in-Action. They suggest that the formalised methodologies that exist currently are not always used in the way they are meant to be used. Instead they are modified and even combined with other methodologies to create a new methodology that fulfils the needs of the organisation. Although not tested in this study, this theory could also be a contributing factor to the low intensity of use of techniques and methods.

9.2.2 Perceived Maturity of IS Departments and the Deployment of Systems Development Methodologies

The findings show that organisations approximately follow the hierarchy of levels as stipulated by the CMM. Although the organisations are all at the Initial CMM level, they do exercise key practices relating to Levels 2, 3 and 4 of the CMM. While the results show a similarity in the hierarchical structure of the clusters, there has been a decline in the application of the CMM key practices since Huisman's study in 1999. This could possibly be due to the fact that organisations have chosen to use other instruments to measure the maturity of their software process or that they are simply not perturbed about the level of maturity of the organisation's IS department.

The sample in this study was smaller than the sample Huisman (1999) had and this could also have contributed to the results showing a decline in the application of CMM key practices.

The CMM level 2 and level 4 practices were found to have no impact on the deployment of systems development methodologies.

A strong relationship was found to exist between the percentage of key practices performed at CMM level 3 and the support for organisational deployment, support for technical design and support for verification and validation. The level 3 key practices were also found to impact on the support for common conception of systems development practice, and the quality, goal achievement and reputation of the IS department.

These results correspond with the findings made by Jiang *et al.* (2004) who also established that project performance is most related to the process engineering and organisational support activities of the CMM Level 3. The lack of significance of Level 2 and 4 key practices with that of deployment further supports the finding by Jiang *et al.* (2004) that the benefits of using the CMM may only be realised once Level 3 maturity is attained, and that the benefits may taper off after Level 3.

9.2.3 Organisational Culture and the Deployment of Systems Development Methodologies

Cluster analysis was performed on the four organisational culture groups as defined by the Competing Values Framework, namely, group culture, developmental culture, hierarchical culture and rational culture. The analysis resulted in two sets of results, one with three clusters and another with four clusters. After interpretation of the results, five distinct clusters emerged: a strong comprehensive culture, a moderate comprehensive culture, a weak hierarchical culture, a weak rational culture and a weak non-developmental culture. The results of the cluster analysis showed that organisations reported a generally weaker affiliation to particular cultural groups than they did in 1999.

The relationship between developmental culture and the deployment of systems development methodologies was the strongest of all the cultural orientations. It has an effect on every perspective of deployment besides support as control technology, support for the evaluation of SD practice and horizontal use. The lack of support for these aspects of deployment can be explained by the inherent nature of this organisational culture group. Developmental culture leans toward decentralisation, differentiation and flexibility. It thus seems logical that it would not advocate the use of any technology across the entire organisation (horizontal use) and also

not render support as a control technology due to its characteristics of decentralisation and flexibility.

There has thus been a dramatic shift in the cultural orientation of organisations' since the study by Huisman was conducted in 1999. In stark contrast to the results obtained in this study, Huisman (1999) found that hierarchical culture influences the deployment of systems development methodologies the greatest. This is interesting since developmental culture and hierarchical culture are direct opposites of each other.

This shift is possibly due to the fact that the *types* of organisations that have entered the information systems development arena are different to those in the past and that they also have a stronger affiliation to the developmental culture.

9.2.4 Organisational Deployment of Systems Development Methodologies

The results revealed that IS department size has a significant negative effect on the deployment of systems development methodologies. In particular, it influences the support aspects of deployment and it also has an effect on the development process and the developed system.

The negative impact is surprising since it is often perceived that large organisations implement systems development methodologies more readily than their smaller counterparts. It is often assumed that the larger the IS department, the greater the support available for processes and tasks associated with software development to be implemented and completed. The results of this study show that this is not always the case.

The support for organisational alignment and horizontal use are impacted by an organisations maturity stage.

Developmental culture was found to be the most influential on deployment of methodologies. This is in stark contrast to the results obtained by Huisman (1999) who found hierarchical culture to have the most impact on the deployment of systems development methodologies.

The time spent on the development of new applications was related to the deployment of methodologies however, no relation was found between deployment and the time spent on planning, analysis and design.

9.2.5 Individual Deployment of Systems Development Methodologies

The results indicated that a very strong relationship exists between the innovation characteristics comprising of relative advantage, complexity (ease of use), compatibility, demonstrability and trialability, and the deployment of systems development methodologies.

Relative advantage, complexity (ease of use), compatibility and trialability have an impact on all the aspects of deployment besides horizontal and vertical use. Demonstrability had a strong relationship with support as organisational alignment, support for technical design and horizontal use.

The impact that compatibility and trialability have on the deployment of systems development methodologies was found to be much stronger than in Huisman (1999).

A developer's education and experience in the use of systems development methodologies does not impact deployment significantly. However, the experience that a developer has in systems development does have an effect on the support for verification and validation. These results are contrary to Huisman (1999) who found that a developer's experience in systems development methodologies impacts the use of methodologies.

No relationship was found between management support and the deployment of systems development methodologies. A positive relationship was found between developer support and the deployment of methodologies. This contrasts with Huisman (1999) who found a positive relationship between management support and deployment and no relationship between developer support and the deployment of systems development methodologies.

The perceived threat to the existence of the IS department is negatively related to the productivity and morale within the IS department.

Voluntariness has a strong negative effect on the support for organisational alignment, support for technical design, and support for verification and validation and horizontal use. Huisman (1999) found no relationship between deployment of methodologies and voluntariness.

Huisman (1999) found that the time spent on planning, analysis and design impacts on the horizontal use, and the time spent on new development impacts the support as cognitive and co-operative technology. These task characteristics had no effect on the deployment of systems development methodologies in this study.

9.3 LIMITATIONS AND FUTURE RESEARCH

9.3.1 Limitations of the study

The sample population that was researched in this study was small despite the effort made to telephonically contact each of the one hundred and thirty six organisations within the ICT Users Handbook. Notwithstanding this, sixty one organisations agreed to participate in the survey. Of the sixty-one organisations, fourteen (23%) completed and returned the questionnaires.

The response of 23% is comparable to previous empirical research conducted in this field (Iivari and Maansaari, 1998; Sharma and Rai, 2000; Jiang *et al*, 2004) where the response rate ranged from 15% to 25%. These response rates were all received to surveys conducted using the postal system.

Prior to the distribution of the questionnaires, Statistical Consultation Services at the university was conferred with and it was stated that the response rates of surveys conducted via email are generally lower than those conducted via the postal system. However, due to time constraints and the problems with the postal system in South Africa at the time of this study, an email survey was favoured.

Although the population researched in this study was smaller than Huisman (1999) and not all of the statistical analyses performed by her could be carried out in this study, accurate comparisons could still be made between the two studies since the appropriate statistical measures were used after receiving advice from Statistical Consultation Services.

9.3.2 Future Research

This study revealed that there has indeed been a change in the use and effectiveness of systems development methodologies since Huisman (1999). An interesting finding was that the intensity of use of methods and techniques has decreased drastically since 1999.

This supports the theory of Avison and Fitzgerald (2003) that the software development industry is undergoing a paradigm shift and moving into a post-methodology era where organisations are not using formal methodologies to develop software anymore. Further research is necessary to confirm this.

Another theory of Fitzgerald *et al.* (2002) called Methods-in-Action could also be a contributing factor to the low intensity of use of methods and techniques. This theory postulates that organisations do not use formalised methodologies in the way they are meant to be used. Instead they are modified or combined with other methodologies to create a new methodology that fulfils the needs of the organisation. This theory was not tested for and there exists a niche for future research into this topic.

The Capability Maturity Model (CMM) was used as a basis for determining the software process maturity of an organisation in this study since Huisman (1999) used the same model. The Capability Maturity Model Integration (CMMI) is currently the new standard for measuring software process maturity in organisations. Further research is required in order to gain insight into the differences between the CMM and CMMI and whether these differences impact the organisational maturity of an organisation and the deployment of systems development methodologies.

A radical change has occurred in the organisational culture within the software development arena. Huisman (1999) found that hierarchical culture had the strongest effect on the deployment of systems development methodologies whereas this study found that developmental culture impacts deployment of methodologies the greatest. These two culture groups are polar opposites and it was speculated in this study that the sudden shift in organisational culture has been caused by the emergence of new types of organisations in the field of software development. Concrete evidence is not available to confirm this and considerable further research is necessary to determine the cause of this sudden change in organisational culture.

BIBLIOGRAPHY

Addison, T.M. and Hamersma, S. (1996) Critical Success Factors for Implementing CASE at a Selection of Companies in South Africa, South African Computer Journal, No. 18, pp. 4-9

Al-Khalifa, K.N. and Espinwall, E.M. (2001) Using the Competing Values Framework to Investigate the Culture of Qatar Industries, Total Quality Management, Vol. 12, No. 4, pp. 417-428

Alavi, M. (1984) An Assessment of the Prototyping Approach to Information Systems Development, Communications of the ACM, June, pp. 556-563

Ambler, S.W. (2001) The Object Primer, Second Edition, Cambridge University Press, Cambridge, England

Aoyama, M. (1998) Web-Based Agile Software, IEEE Software, November/December, pp. 56-65

Avison, D.E. and Fitzgerald, G. (1995) Information Systems Development: Methodologies, Techniques and Tools, McGraw-Hill Publishing Company, Berkshire, England

Avison, D.E. and Fitzgerald, G. (2003) Where Now for Development Methodologies?, Communications of the ACM, Vol. 46, No. 1, pp. 79-82

Avison, D.E. and Fitzgerald, G. (2002) Information Systems Development: Methodologies, Techniques, and Tools, Third Edition, McGraw-Hill Publishing Company, London, England

Bachmann, D. and Elfrink, J. (1996) Tracking the Progress of E-mail versus Snail-Mail. MarketingResearch, Vol. 8, No. 2, pp. 31-35.

Bansler, J. and Bodker, K. (1993) A Reappraisal of Structured Analysis: Design in an Organisational Context, ACM Transactions on Information Systems, Vol. 11, No.2, pp. 165-193

Baskerville, R., Travis, J. and Truex, D. (1992) Systems without Method: The Impact of New Methodologies on Information Systems Development Projects, in Kendall, K.E., Lyytinen, K. and DeGross, J.I. (eds), The Impact of Computer Supported Technologies on Information Systems Development, Amsterdam, pp. 241-269

Bergland, G. (1981) A Guided Tour of Program Design Methodologies, IEEE Computer, October, pp. 13-37

Beynon-Davies, P (1998) Information Systems Development, Third Edition, An Introduction to Information Systems Engineering, Macmillan, London

Beynon-Davies, P. and Williams, M.D. (2003) The Diffusion of Information Systems Development Methods, Journal of Strategic Information Systems, Vol. 12, pp. 29-46

Boehm, B. (1981) Software Engineering Economics, Prentice Hall, Eaglewood Cliffs, New Jersey

Bostrom, R.P. and Heinen, J.S. (1977) MIS Problems and Failures: A Socio-Technical Perspective, MIS Quarterly, Vol. 1, No. 3, pp. 17-32

Brandl, D. (2005) Control Engineering, Software Development Capability, Vol. 52, No. 8, pp. 20-20

Brinkkemper, S. (1996) Method Engineering: Engineering of Information Systems Development Methods and Tools, Information and Software Technology, Vol. 38, pp. 275-280

Brooks, F. (1987) No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer Magazine, April, pp. 10-19

Brown, A.W. (2000) Large-Scale Component-Based Development, Prentice Hall, Upper Saddle River, New Jersey

Butler, K.L. (1995) The Economic Benefits of Software Process Management, Cross Talk, July, pp. 14-17

- Canning, R. (1956) in Agresti, W. (1986) *New Paradigms for Software Development*, IEEE Computer Society Press, Washington DC
- Chatzoglou, P.D. and Macaullay, L.A. (1996) *Requirements Capture and IS Methodologies*, Information Systems Journal, Vol. 6, pp. 209-225
- Clements, P.C. (1995) *From Subroutines to Subsystems: Component-Based Software Development*, American Programmer, Vol. 8, No. 11
- Coad, P. and Yourdon, E. (1991) *Object-Oriented Analysis*, 2nd Edition, Yourdon Press, New Jersey
- Colter, M. (1982) *Evolution of the Structured Methodologies*, in Couger et al., (1982) *Advanced System Development Feasibility Techniques*, Wiley and Sons, New York, pp 73-96
- Conger, S. (1994) *The New Software Engineering*, Wadsworth Publishing Company, Belmont, California
- Connors, D.T. (1992) *Software Development Methodologies and Traditional and Modern Information Systems*, Software Engineering Notes, Vol. 17, No. 2, pp. 43-49
- Couper, M. P. (2000) *Web-Based Surveys: A Review of Issues and Approaches*, Public Opinion Quarterly, Vol. 64, No. 4, pp. 464-494.
- Crnkovic, I. and Larsson, M. (2002) *The Journal of Systems and Software*, No. 61, pp. 201-212
- Dekleva, S and Drehmer, D. (1997) *Measuring Software Engineering Evolution: A Rasch Calibration*, Information Systems Research, Vol. 8, No. 1, pp. 95-104
- DeMarco, T. (1978) *Structured Analysis and System Specification*, Yourdon Press, New Jersey
- Denison, D.R. and Spreitzer, G.M. (1991) *Organisational Culture and Organisational Development: A Competing Values Approach*, Research in Organisational Change and Development, Vol. 5, pp. 1-21

Dietrich, G.B., Walz, D.B., Wynekoop, J.L. (1997) The Failure of SDT Diffusion: A Case for Mass Customisation, IEEE Transactions on Engineering Management, Vol. 44, No. 4, pp. 390-398

Dion, R. (1993) Process Improvement and the Corporate Balance Sheet, IEEE Software, Vol. 10, pp. 14-17

English, A. (2002) Extreme Programming: It's Worth a Look, IT Professional, Vol.4, No.3, pp. 48-50

Erlank, S., Pelteret, D. and Meskin, M. (1991) An Analysis of the Usage of System Development Methods in South Africa, South African Computer Journal, No. 4, pp. 68-77

Eva, M. and Guilford, S. (1996) Committed to a RADical Approach? A Survey of Systems Development Methods in Practice, in Jayaratna, N. and Fitzgerald, B., (eds.) Lessons Learned from the Use of Methodologies: Fourth Conference on Information Systems Methodologies, pp. 87-96

Ewusi-Mensah, K. (1997) Critical Issues in Abandoned Information Systems Development Projects, Communications of the ACM, Vol. 40, No. 9, pp. 74-80

Fayad, M.E. and Schmidt, D.C. (1997) Object-Oriented Application Frameworks, Communications of the ACM, Vol. 40, No. 10, pp. 32-38

Fichman, R.G. (1992) Information Technology Diffusion: A Review of Empirical Research, in DeGross, J.I., Becker, J.D. and Elam, J.J. (eds.) Proceedings of the Thirteenth International Conference on Information Systems, Dallas, TX, pp. 195-206

Finkelstein, C. (1989) An Introduction to Information Engineering, Addison-Wesley, Sydney

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

Fitzgerald, B., Russo, N.L. and Stolterman, E. (2002) Information Systems Development: Methods in Action, McGraw-Hill, Maidenhead, Berkshire

Gallivan, M. and Strite, M. (2005) Information Technology and Culture: Identifying Fragmentary and Holistic Perspectives of Culture, Information and Organisation, Vol. 15, No. 4, pp. 295-338

Gane, C. (1991) The Triumph of the Structured Movement, American Programmer, Vol. 4, No. 11, pp. 38-39

Garton, L., Haythornthwaite, C. and Wellman, B. (1999) Studying On-Line Social Networks in S. Jones (Ed.), Doing Internet Research: Critical Issues and Methods for Examining the Net pp. 75-105. Thousand Oaks, CA: Sage

Glass, R.L. (2003) A Mugwump's-Eye View of Web Work, Communications of the ACM, Vol. 46, No. 8, pp. 21-23

Good, J.M. (2003) A Pragmatic Approach to the Implementation of Agile Software Development Methodologies in Plan-Driven Organisations, Lincoln University

Haines, C.G., Carney, D. and Foreman, J. (1997) Component-Based Software Development/COTS Integration, Software Technology Review, http://www.sei.cmu.edu/str/descriptions/CBD_body.html

Hardy, C.J., Thompson, J.B. and Edwards, H.M. (1995) The Use, Limitations and Customization of Structured Systems Development Methods in the United Kingdom, Information and Software Technology, Vol. 37, No. 9, pp. 467-477

Hauser, M. (1998) Organisational Culture and Innovativeness of Firms – An Integrative View, International Journal of Technology Management, Vol. 16, No. 1, pp. 239-255

Henson, K. and Hughes, C. (1991) A Two-Dimensional Approach to Systems Development, Journal of Information Systems Management, Winter Edition, pp. 35-43

Herbsleb, J.D. and Goldenson, D.R. (1996) A System Survey of CMM Experience and Results, Proceedings of ICSE, Vol. 18, pp. 323-330

Hopkins, J. (2000) Component Primer, Communications of the ACM, Vol. 43, No. 10, pp. 27-30

- Hughes, J. (1998) Selection and Evaluation of Information Systems Methodologies: The Gap between Theory and Practice, IEEE Proceedings on Software, Vol. 145, No. 4, pp. 100-104
- Huisman, H.M. (1999) The Deployment of Systems Development Methodologies: A South African Experience, Thesis – PhD
- Hull, M.E.C., Taylor, P.S., Hanna, J.R.P. and Millar, R.J. (2002) Software Development Process – An Assessment, Information and Software Technology, No. 44, pp. 1-12
- Hult, G.T.M., Ketchen, D.J. and Nichols, E.L. (2002) An Examination of Cultural Competitiveness and Order Fulfilment Cycle Time with Supply Chains, Academy of Management Journal, Vol. 53, No. 3, pp. 557-586
- Humphrey, W.S. (1989) Managing the Software Process, Addison-Wesley, Reading, Massachusetts
- Humphrey W.S., Snyder, T.R. and Willis, R.R. (1993) Software Process Improvement at Hughes Aircraft, IEEE Software, Vol. 8, No. 4, pp. 11-23
- Iivari, J. (1993) From a Macro Innovation Theory of IS Diffusion to a Micro Innovation Theory of IS Adoption: An Application to CASE Adoption, in Avison, D., Kendall, J.E. and DeGross, J.J. (eds.) Human, Organisational and Social Dimensions of Information Systems Development, Elsevier Science Publishers B.V., North-Holland, pp. 295-320
- Iivari, J., Hirscheim, R. and Klein, H.K. (1998) A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies, Information Systems Research, Vol. 9, No. 2, pp.164-193
- Iivari, J., Hirscheim, R. and Klein, H.K. (1999) Beyond Methodologies: Keeping up with Information Systems Development Approaches and Methodologies, Information Systems Research, Vol. 9, No. 2, pp. 164-193
- Iivari, J. and Maansaari, J. (1998) The Usage of Systems Development Methods: Are We Stuck to Old Practices? Information and Software Technology, Vol. 40, pp. 501-510

- Jiang, J.J., Klein, G., Hwang, H., Huang, J. and Hung, S. (2004) An Exploration of the Relationship between Software Development Process Maturity and Project Performance, *Information & Management*, Vol. 41, pp. 279-288
- Joshi, J.B.D., Aref, W.G., Ghafoor, A. and Spafford, E.H. (2001) Security Models for Web-Based Applications, *Communications of the ACM*, Vol. 44, No. 2, pp. 38-44
- Kaarst-Brown, M.L. and Robey, D. (1999) More on Myth, Magic and Metaphor: Cultural Insights into the Management of Information Technology in Organisations, *Information Technology and People*, Vol. 12, No. 2, pp. 192-217
- Kaarst-Brown, M.L., Nicholson, S., von Dran, G.M. and Stanton, J.M. (2004) Organisational Cultures of Libraries as a Strategic Resource, *Library Trends*, Vol. 53, No. 1, pp. 33-53
- Kay, R. (2005) CMMI, *Computerworld*, Vol. 39, No. 4, pp. 28-28
- Kelly, S. and Rossi, M. (1998) Evaluating Method Engineer Performance: An Error Classification and Preliminary Empirical Study, *Australian Journal of Information Systems*, Vol. 6, No. 1, pp. 39-47
- Kwon, T.H. and Zmud, R.W. (1987) Unifying the Fragmented Models of Information Systems Implementation, in Boland R.J. and Hirschheim, R.A. (eds.) *Critical Issues in Information Systems Research*, John Wiley & Sons, New York, pp. 227-251
- Lai, V.S. and Guynes, J.L. (1997) An Assessment of the Influence of Organisational Characteristics on Information Technology Adoption Decision: A Discriminative Approach, *IEEE Transactions on Engineering Management*, Vol. 44, No. 2, pp. 146-157
- Lee, D.T. (1987) Computer Information System Development Methodologies: A Comparative Analysis, *AFIPS Conference Proceedings*, Vol. 56, pp. 683-692
- Lee, S.C. and Shirani, A.I. (2004) A Component-Based Methodology for Web Application Development, *The Journal of Systems and Software*, No. 71, pp. 177-187

- Lippert, M. and Roock, S. (2001) Adapting XP to Complex Application Domains, ACM, 2001, pp. 316-317
- Llieva, J., Baron, S. and Healey, N. M. (2002) Online Surveys in Marketing Research: Pros and Cons, *International Journal of Market Research*, Vol. 44, No. 3, pp. 361-367.
- Louis, M.R. (1990) Acculturation in the Workplace: Newcomers as Lay Ethnographers, in B. Schneider (ed.), *Organisational Climate and Culture* pp. 85-129, San Francisco, CA: Jossey-Bass
- Lyytinen, K., Mathiassen, L. and Ropponen, J. (1996) A framework for software risk management. *Journal of Information Technology*, Vol. 11, No. 4, pp. 275–287
- Lyytinen, K., Mathiassen, L. and Ropponen, J. (1998) Attention Shaping and Software Risk: A Categorical Analysis of Four Classical Approaches, *Information Systems Research*, Vol. 9, No. 3, pp. 233-255
- Markus, M.L. (1983) Power, Politics and MIS Implementation, *Communications of the ACM*, Vol. 26, pp. 430-444
- Markus, M.L. and Robey, D. (1983) The Organisational Validity of MIS, *Human Relations*, Vol. 36, pp. 203-226
- Martin, J. and Finkelstein, C. (1981) *Information Engineering*, Savant Institute, UK
- Martin, J. (1982) *Strategic Data-Planning Methodologies*, Prentice Hall, Engelwood Cliffs
- Martin, J. and Odell, J. (1992) *Object Oriented Analysis and Design*, Prentice-Hall, Englewood Cliffs
- Mathiassen, L., Munk-Madsen, A., Nielsen, P.A. and Stage, J. (2000) *Object Oriented Analysis and Design*, Marko Publishing ApS, Aalborg, Denmark
- McChesney, I.R. and Glass, D. (1993) Post-Implementation Management of CASE Methodology, *European Journal of Information Systems*, Vol. 2, No. 3, pp. 201-209

- Mehta, R. and Sivadas, E. (1995) Comparing Response Rates and Response Content in Mail versus Electronic Mail Surveys, *Journal of the Market Research Society*, Vol. 37, No. 4, pp. 429-439.
- Michalisin, M.D., Smith, R.D. and Kline, D.M. (1997) In Search of Strategic Assets, *International Journal of Organisational Analysis*, Vol. 5, No. 4, pp. 360-387
- Mili, R., Mili, A., and Mittermeir, R.T. (1997) Storing and Retrieving Components: A Refinement-Based System, *IEEE Transactions on Software Engineering*, Vol. 23, No. 7, pp. 445-460
- Moore, G.C. and Benbasat, I (1991) Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation, *Information Systems Research*, Vol. 2, No. 3, pp. 365-395
- Murugesan, S. (1999) Editorial, *ACM SIGWEB Newsletter*, No. 3, pp. 3
- Munro, J. (2003) Extreme Programming, *PC Magazine*, Vol. 22 Issue 3, pp. 68
- Nandhakumar, J. and Avison, D.E. (1999) The Fiction of Methodological Development: A Field Study of Information Systems Development, *Information Technology & People*, Vol. 12, No. 2, pp. 176-191
- Newkirk, J. and Martin, C. (2000) Extreme Programming in Practice, *ACM*, pp. 25-26
- Nuckols, C. and Canna, J. (2003) eXtreme Documentation, *Intercom*, February, pp. 6-9
- Orr, K. (1977) *Structured Systems Development*, Yourdon Press, New York
- Orr K. (1989) Methodology: The Experts Speak, *BYTE*, April, pp. 221-233
- Palvia, P. and Nosek, J.T. (1993) A Field Examination of System Life Cycle Techniques and Methodologies, *Information and Management*, Vol. 25, pp. 73-84

Parnas, D. (1972) On the Criteria to be used in Decomposing Systems into Modules, Communications of the ACM, Vol. 15, No. 12, pp. 1053-1058

Paulk, M., Curtis, B., Chrissis, M.B. and Weber, C.V. (1993) Capability Maturity Model for Software: Version 1.1, IEEE Software, Vol. 10, No. 6, pp. 18-27

PC Week (1995), No. 16, pp. 68

Prescott, M.B. and Conger, S.A. (1995) Information Technology Innovations: A Classification by IT Locus of Impact and Research Approach, The DATABASE for Advances in Information Systems, Vol. 26, No. 2/3, pp. 20- 42

Punter, T. and Lemmen, K. (1996) The MEMA- Model: Towards a New Approach for Method Engineering, Information and Software Technology, Vol. 38, pp. 295-305

Quinn, R.E. and Rohrbaugh, J. (1983) A Spatial Model of Effectiveness Criteria: Towards a Competing Values Approach to Organisational Effectiveness, Public Productivity Review, Vol. 5, pp. 122-140

Quinn, R.E. and Cameron, K.S. (1983) Organisational Lifecycles and Shifting Criteria of Effectiveness, Some Preliminary Evidence, Management Science, Vol. 29, pp. 33-51

Quinn, R.E. and Kimberly, J.R. (1984) Paradox, Planning and Perseverance: Guidelines for Managerial Practice, in Kimberly, J.R. and Quinn, R.E. (eds.) New Futures: The Challenge of Managing Organisational Transitions, Dow Jones-Irwin, Homewood, ILL, pp. 295-313

Quinn, R.E. and Cameron, K.S. (1999) Diagnosing and Changing Organisational Culture: Based on the Competing Values Framework, Reading, MA: Addison-Wesley

Rai, A. and Al-Hindi, H. (2000) The Effects of Development Process Modelling and Task Uncertainty on Development Quality Performance, Information & Management, Vol. 37, pp. 335-346

Rational Software Corporation (1999) Rational Unified Process: Best Practices for Software Development Teams

Reich, B.H. and Benbasat, I. (2000) Factors that Influence the Social Dimension of Alignment between Business and Information Technology Objectives, MIS Quarterly, Vol. 24, No. 1, pp. 81-113

Reifer, D.J. (2002) How Good are Agile Methods? IEEE Software, Vol. 19, No. 4, pp. 16-18

Retalis, S., Psaromiligkos, Y. and Avgeriou, P. (2000) Web Engineering: New Discipline, New Educational Challenges, Information Services & Use, No. 20, pp. 95-108

Robey, D. and Azevedo, A. (1994) Cultural Analysis of the Organisational Consequences of IT, Accounting, Management and Information Technology, Vol. 4, No. 1, pp. 23-37

Rogers, E.M. (1995) Diffusions of Innovations, Fourth edition, The Free Press, New York

Royce, W. (1970) Managing the Development of Large Software Systems, Proceedings of IEEE Wescon

Russo, N., Wynekoop, J. and Walz, D. (1995) The Use and Adaptation of Systems Development Methodologies, in Khosrowpour, M. (ed.), Managing Information and Communications in a Changing Global Environment, Idea Group Publishing, PA

Saeki, M. (1998) A Meta-Model for Method Integration, Information and Software Technology, Vol. 39, pp. 925-932

Schach, S.R. (1997) Software engineering with Java, Richard D. Irwin, USA

Schein, E.H. (1985) Organisational Culture and Leadership: A Dynamic View, San Francisco, CA, Josey-Bass

Schwalbe, K. (2000) Information Technology Project Management, Course Technology, New York

Senn, J.A. and Wynekoop, J.L. (1995) The Other Side of CASE Implementation, Information Systems Management, Vol. 12, No. 4, pp. 7-14

- Sharma, S. and Rai, A. (2000) CASE Deployment in IS Organisations, Communications of the ACM, Vol. 43, No. 1, pp. 80-88
- Shine Technologies (2003) Agile Methodologies – Survey Results, Shine Technologies Pty Ltd, <http://shinotech.com/ShineTechAgileSurvey/2003-01-17.pdf>
- Siau, K. and Rossi, M. (1998) Evaluating Information Modelling Methods: A Review, HICSS-31, IEEE Computer Society, Big Island, HI
- Stanton, J. M. (1998) An Empirical Assessment of Data Collection Using the Internet, Personnel Psychology, Vol. 51, No. 3, pp. 709-725.
- Storck, J. and Hill, P.A. (2000) Knowledge Diffusion through “Strategic Communities”, Sloan Management Review, Vol. 41, No. 2, pp. 63-74
- Sumner, M. and Sitek, J (1986) Are Structured Methods for Systems Analysis and Design Being Used? Journal of Systems Management, pp.18-23
- Sumner, M (1992) The Impact of Computer-Assisted Software Engineering on Systems Development, in Kendall, K.E. et al. (eds.) The Impact of Computer Supported Technologies on Information systems Development, Elsevier Science Publishers B.V., North-Holland, pp. 43-60
- Swanson, E.B. (1994) Information Systems Innovation among Organisations, Management Science, Vol. 40, No. 9, pp. 1069-1092
- Szyperski, C. (1998) Component Software: Beyond Object-Oriented Programming, ACM Press, New York
- Taylor, H. (2000) Does Internet Research Work? Comparing Electronic Survey Results with Telephone Survey, International Journal of Market Research, Vol. 42, No. 1, pp. 51-63
- Teichrow, D. and Sayani, H. (1971) Automation of System Building. In Couger et al. (1982) Advanced System Development Feasibility Techniques, Wiley and Sons, New York, pp. 437-446

The Standish Group (1995) Chaos, Standish Group Report

Thomann, J. (1994) Data Modelling in an OO World, American Programmer, Vol. 7, No. 10, pp. 44-53

Tolvanen, J.P. (1995) Incremental Method Development for Business Modelling: An Action Research Case Study, Proceedings of 6th Workshop on Next Generation CASE Tools, Grosz, G. (ed.), pp. 79-98, University of Paris, Paris

Tolvanen, J.P., Rossi, M. and Liu, H. (1996) Method Engineering: Current Research Directions and Implications for Future Research in: Method Engineering, Principles of Method Construction and Support. Brinkkemper, S., Lyytinen, K. and Welke, R. (eds.), pp. 296-317. Chapman & Hall Publications, New York

VanDenKerkhof, E.G., Parlow, J.L., Goldstein, D.H. and Milne, B. (2004) In Canada, Anesthesiologists are less likely to Respond to an Electronic, Compared to a Paper Questionnaire, Canadian Journal of Anesthesia, Vol. 51, No. 5, pp. 449-454

Van de Velde, R. (1992) CASE Adoption at Imperial Oil, Journal of Systems Management, Vol. 43, No. 9, pp. 24, 25, 28, 29, 36, 38

Veryard, R. (1985) What are Methodologies Good For? Data Processing, Vol. 27, No. 6, pp. 9-12

Vickers, M.H. (1999) Information Technology Development Methodologies: Towards a Non-Positivist, Developmental Paradigm, The Journal of Management Development, Vol.18, No. 3, pp. 255-272

Vitharana, P., Zahedi, F. and Jain, H. (2000) Strategy-Based Design of Reusable Business Components, University of Wisconsin-Milwaukee, School of Business Administration, Working Paper

Vitharana, P., Zahedi, F. and Jain, H. (2003) Design, Retrieval and Assembly in Component-Based Software Development, Communications of the ACM, Vol. 46, No. 11, pp. 97-102

Vitharana, P. (2003) Risks and Challenges of Component-Based software Development, Communications of the ACM, Vol. 46, No. 8, pp. 67-72

Walters, S.A., Broady, J.E. and Hartley, R.J. (1994) A Review of Information Systems Development Methodologies, Library Management, Vol. 15, No. 6, pp. 5-19

Warnier, J. (1976) Logical Construction of Programs, Van Nostrand Reinhold, New York

Wikipedia (2005) Capability Maturity Model,
http://en.wikipedia.org/wiki/Capability_Maturity_Model

Woodman, R.W. and Pasmore, W.A. (eds.) (1991) Research in Organisational Change and Development, JAI Press, Greenwich, CT

Wynekoop, J.L., and Russo, N.L. (1993) Studying Systems Development Methodologies: Unanswered Questions and the Research Gap, in DeGross, J.I. et al. (eds.) Proceedings of the Fourteenth International Conference on Information Systems, Orlando, FL, pp. 181-190

Wynekoop, J.L., and Russo, N.L. (1997) Studying Systems Development Methodologies: An Examination of Research Methods, Information Systems Journal, Vol. 7, pp. 47-65

Yourdon, E. (1979) Classics in Software Engineering, Yourdon Press, New York

Yourdon, E. (1988) Sayonara, Structured Stuff, American Programmer, Vol. 1, No. 11, pp. 40-49

Yourdon, E. (1991) Sayonara, Once Again, Structured Stuff, American Programmer, Vol. 4, No. 8, pp. 31-38

Yun, G. W. and Trumbo, C. W. (2000) Comparative Response to a Survey Executed by Post, Email, and Web Form. Journal of Computer-Mediated Communication, Vol. 6, No. 1, pp. 107-127

Zammuto, R.F. and Krakower, J.Y. (1991) Quantitative and Qualitative Studies of Organisational Culture, Research in Organisational Change and Development, An Annual Series Featuring Advances in Theory, Methodology, and Research, Vol. 5, pp. 83-114