UNBOUNDED RULE-BASED EXPERT SYSTEM FOR SELECTING SOFTWARE DEVELOPMENT METHODOLOGIES

BY

MACHEQUE VHUTSHILO

STUDENT NUMBER: 11612881

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF MASTERS OF COMMERE IN BUSINESS INFORMATION SYSTEMS

SCHOOL OF MANAGEMENT SCIENCES

UNIVERSITY OF VENDA

SUPERVISOR: Prof A Kadyamatimba …… Date :……

CO-SUPERVISORS: Prof N.M Ochara: …… Date :……

CO-SUPERVISORS: Mr. D. Tutani:… …… Date :……..

2019

DECLARATION

I Macheque Vhutshilo, hereby declare that this dissertation for the Masters of Commerce in Business Information Systems submitted to the Department of Business Information Systems at the University of Venda has not been submitted previously for any degree at this or another university. It is my original in design and execution, and all reference material contained therein has been duly acknowledged.

Student ……... Date ……

ABSTRACT

The extent of success of a given project can be increased by using an appropriate Project Management Methodology (PMM) that takes into account the specific characteristics of the project (such as complexity, size, budget, nature of risk, etc.). PMMs have evolved over the years to become more diverse, complex, with evolving and dynamic ICT platforms. Such PMMs have traditionally been used as frameworks to guide the project management process for decision makers (such as Project Managers, Project Owners and Project Teams). The choice of selecting an appropriate project methodology is daunting; apart from other considerations related to project characteristics such as budget, scope, schedule, performance and resource constraints. One of the vital stages of a successful software development project is selecting a good software development methodology that best suits that project.

The aim of this research is to investigate the critical factors to be considered by project managers in the selection of the software development methodology for the project. These critical factors are then used as a foundation for an architecture for an “unbounded rule-based expert system. A survey was conducted amongst project managers to determine the critical factors necessary for the selection of a software development methodology. From the findings of the study, it was established the critical factors revolved around three constructs of Project Excellence Enablers, Excellent Project Management Practices, and Business Value Proposition factors. The findings from this study therefore provided a rationale and a basis for the evolution of an “Unbounded Rule-Based Expert Systems Architecture” as a basis for the selection of the right software development methodology

Keywords: Software Development Methodology, Expert systems, Artificial intelligence, Rule-based Expert system.

ACKNOWLEDGEMENTS

I would like to thank God for granting me His mercy and wisdom to accomplish this work successfully. The deeds of God's hands are faithful and just and His precepts are trustworthy (Psalm 111:7). Truly God is awesome, and His promises are "Yes and Amen" (2 Corinthians 1:20). I will always love and serve Him with all my heart. I also would like to extend my special thanks to Prof A Kadyamatimba, Prof Ochara NM, Mr. Donald Tutani and my family they have always believed in me, even when my enthusiasm was at its lowest. During difficult times, they would be supportive to me and remind me of the fruits of perseverance in the face of hardship, and eventually rekindle my ambition. I am grateful to God for blessing me with such family whose hearts are deeply committed to Him. I am grateful to God for them being part of my life.

My great appreciation also goes to all the companies that were involved in this research project; my appreciation to their employees for the openness, time and effort they applied in expressing their views. Thanks are also due to my colleagues motivations have always made me remember that nothing is impossible with God and that for every pain there is a reward. My persistence and strength in this endeavour were as a result of their prayers and their availability in times of distress.

As to myself, I am deeply grateful to God to for being His child whom He loves so much through Jesus Christ. I pray that all the fruits of the Spirit according to Galatians 5:22 (But the fruit of the Spirit is love, joy, peace, patience, kindness, goodness, faithfulness, gentleness, self-control) be apparent in my life, and that He may grant me the wisdom and power to pursue my career goals and His purpose for my life.

To all, I deeply and humbly express my thanks for the job well done. Keep up with the good work and May God Bless You.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI - Artificial intelligent

CMS - Centres for Medicare and Medicaid Services

EA -Intelligent Agent

EP -Excellent PM Practices

EPMP -Excellent Project Management Practices

ES - Expert System

JSD - Jackson Systems Development

KM -Knowledge Management

PCA - Principal Component Analysis

PEE -Project Excellence Enablers

PM - Project Excellence Enablers

PMM -Project Management Methodology

RAD -Rapid Application Development

RAD- Rapid Application Development

RE -Real Expectations

RUP -Rational Unified Process

SDLC – Software Development Life cycle

SDMs – Software Development Methodologies

SPSS -The Statistical Package for Social Sciences

SSADM - Structured Systems Analysis and Design Method

TCA -Thematic Content Analysis

URB-ES Unbounded Rule-Base expert system

XP - Extreme programming

CHAPTER 1: INTRODUCTION

1.1 Background to the Study

Software has undoubtedly become the most valuable and sought-after asset in most organisations in the world, and thus the demand for software has also grown tremendously. This high demand of software has become a problem for software developers as they have to meet deadlines while producing high-quality software. When initiating a software development, it is very important to use a methodology that increases success rate to meet the deadlines, minimises cost and fulfils the correct given requirements (Joussen, 2004).

Dalal (2011) noted that more projects were failing and less successful to support his statement he quoted from The Standish Group Chaos Report 2009 which indicated the following: “the results show a marked decrease in project success rates, with 32% of all projects succeeding which are delivered on time, on budget, with required features and functions" says Jim Johnson, chairman of The Standish Group, "44% were challenged which are late, over-budget, and/or with less than the required features and functions and 24% failed, which were cancelled prior to completion or delivered and never used. These numbers represent a downtick in the success rates from the previous study, as well as a significant increase in the number of failures", says Jim Crear, Standish Group CIO, "They are low point in the last five study periods. This year's results represent the highest failure rate in over a decade."

Young (2013) asserts that a software development methodology is a way of managing a software development project. It therefore, addresses issues like selecting features for inclusion in the current version, when software will be released, who works on what, and what testing is done. The use of critically adequate software development methodology for a given project embarked on will play a valuable role in ensuring that the software product is produced and delivered on schedule, within cost and meets users’ requirements (Christenson, 2007). The biggest challenge is that developers have a very large pool of software development methodologies to select from. The one-size fits all theory does not apply in this case, because the choice is determined by the project at hand. Thus, the developer or project manager needs to be well-equipped with skills of how to choose which methodology is best suitable for the project at hand.

According to Zaeid et al., (2013), there is no methodology that can be described as the best as it all depends on the project at hand. The project characteristics will determine which development methodologies will best suit the project. There has to be a domain expert to analyse a project`s properties then figure out what process is to be followed. There is a need

to automate this difficult task since certain properties of a project always determine the best methodology to use. This research contains an analysis and design of a rule-based expert system that replicates the reasoning of an expert responsible for selecting a software development methodology. According to Buchanan and Duda (1983), a rule based expert system is a computer program that provides expert-level solutions to 'important problems and is:

- **Heuristic** -- i.e., it reasons with judgmental knowledge as well as with formal knowledge of established theories;
- **Transparent** -- i.e., it provides explanations of its line of reasoning and answers to queries about its knowledge; and
- **Flexible** -- i.e., it integrates new knowledge incrementally into its existing store of knowledge.

Artificial intelligence is concerned with the development of computers able to engage in human-like thought processes such as learning, reasoning, and self-correction (Kok et al., 2009).

1.2 Statement of the Problem[1]

The extent of success of a given software development project can be increased by using an appropriate methodology that suits the specific characteristics of that project. Over time, a wide range of software development methodologies have been developed, and project managers have a daunting task of selecting a suitable methodology for a project (Geambaşu, Jianu, Jianu & Gavrilă, 2011). According to the European Commission Directorate General for Regional Policy and Cohesion (2014), the use of an adequate methodology for each project plays a valuable role in assuring that the software is delivered on schedule, within cost and meets users' requirements. Therefore, there is a need to assist developers in making the correct choice which can be achieved by researching and developing a software agent that uses internal rules to make the correct choice for a project based on the project characteristics. One of the biggest challenges for developers is the process of selecting a software development methodology that best suits a project. This research focuses on how to best design a Rule-based system that solves the challenges that software developers are facing.

[1] Preliminary work of this dissertation was done as an unpublished honours project titled, "Intelligent Agent to Select Software Development Methodology that Best Suits a Given Project". This dissertation advances from the previous work to emphasize the architecture of unbounded rule-based systems. Further, this dissertations develops the architecture into a rule-based expert system.

1.3 Aim, Objectives and Research Questions of the study

The aim, objectives and research questions are given in the following subsections 1.3.1, 1.3.2 and 1.3.3 respectively:

1.3.1 Aim of the Study

The aim of this study was to investigate the architecture of Rule-based expert systems in order to develop an unbounded Rule-based expert system.

1.3.2 Objectives of the Study

The research objectives are the following:

i. Investigate the architecture of Rule-based expert software.
ii. Identify the key factors that influence the decision of choosing an adequate software development methodology.
iii. To develop an Unbounded Rule-based expert system software prototype.

1.3.3 Research Questions

The research question for the study are given as follows:

i. What is the architecture of Rule-based expert software?
ii. What are the key factors that influence the decision of choosing an adequate software development methodology?
iii. How develop software prototype of an Unbounded Rule-based expert system.

1.4 Significance of the proposed study

The proposed expert system will contribute to the software development body of knowledge. It could also benefit program managers in making intelligent decisions and reducing work load. The study uses a unique mixed research method that combines qualitative method and design and creation method to create an unbounded rule-base expert system for selecting software development methodologies. These methodologies may be used by other researchers to develop their own system.

1.5 Delimitations of the study

The study will be delimited to Johannesburg and Pretoria because that is where there are more companies that deal with software development and for this study the researcher is targeting project managers and software developer.

1.6 Limitations of the study.

There are various constrain that will limit the success of study and these include the following:

- Financial constrain
- The time limit constrains
- Limited internet access

1.7 Operational Definitions

The following are the definitions of the terms:

- **Software Development Methodology** is a framework that is used to structure, plan, and control the process of developing an information system (Zaeid, 2013).
- **Expert system** is a computer program which is designed to carry out tasks associated with a human know-how? It mimics judgments and decision making of human being (Buchnan & Duda, 1983).
- **Artificial intelligence** is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans (Zaeid, 2013).
- **Rule-based Expert system is** a set of "if- then" statements that uses a set of assertions, to which rules on how to act upon those assertions are created (Buchnan & Duda, 1983).

1.8 The research layout

The chapters of the dissertation are outlined as shown below consequently and the brief description of chapters is provided.

Chapter 1 Introduction

This presents the problem statement, research questions, and research objectives, significance of study, delimitations and assumptions made for the rule-based expert system development, previous similar study findings and the structure of the dissertation.

Chapter 2 Literature Review

This chapter presents a review of research documents that provide theory on expert systems, components of existing rule-based systems and how they relate to the general architecture of expert systems. The structure of this chapter has been arranged chronologically so that the

reader can easily follow the role of critical success factors and SDMs towards IT project success.

Chapter 3 Research Methodology

The research design and research methodology are presented. The selection of the Software Development Methodology is also described and justified.

Chapter 4 results analysis

The purpose of this chapter is to interpret and discuss the results of the analysis according to the research questions in order to fulfil the aims and objectives of this study and presents the detailed design of the proposed rule-based system prototype.

Chapter 5 Design and Implementation

This chapter presents the technical implementation of the URB-ES system. Interfaces and code snippets where presented and explained, also described is the Use Case and Activity diagram of the system. A brief outline of how testing was carried out was also given

Chapter 6 Conclusion and recommendation of future work.

This chapter presents the findings of this studies

1.9 Summary

Chapter one presented the background of the research, objectives, motivation and steps taken to achieve the objectives and finally the dissertation outline is given. In the next chapter, the researcher presents a literature review, the study and analysis of research documents that helped in the process of building important concepts in the subject under study.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Much of research has been done in the field of software engineering in terms of how to develop an expert system to produce the best results. This research focused on the designing and developing of an Intelligent Agent (EA) to select a Software Development Methodology expert system that will help developers select the best software development methodology for a particular project at hand.

2.1.1 Structure of an expert system

According to David (1994), an Expert System (ES) is a computer program which is designed to carry out tasks associated with a human expert. It is a program which tries to do things which are usually regarded as the province of human, involving judgments and decision-making. As human experts tend to have a great deal of specialized knowledge in their fields, ESs are usually knowledge-based systems which contain large database of knowledge information. Figure 2.1 shows the components (User interface, Knowledge base and Interference engine) of an ES which are briefly explained below.

i. The user interface

This is the component that allows a non-expert user to query (question) the expert system, and to receive advice. The user-interface is designed to be as simple to use as possible. The user interface of an expect system is the means of communication between a user wishing to solve a problem and the problem solver the (expect system). It is vital that this communication is as meaningful and useful as possible. Otherwise, the expert system will neither be useful nor used. The user interface may be command driven, event driven and icon oriented.

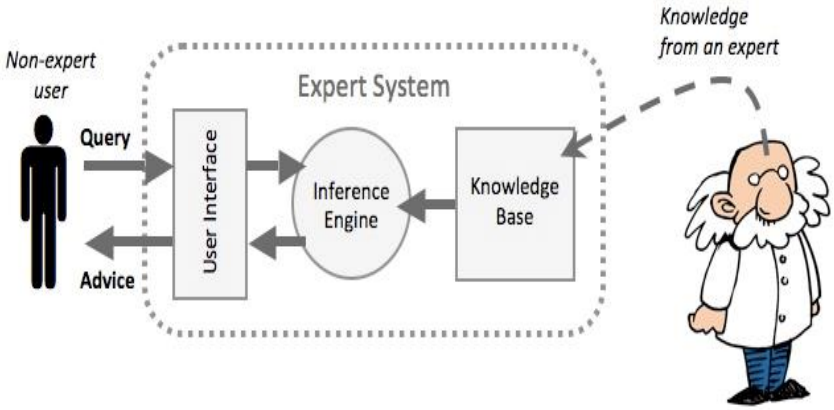


***Figure 2.1**: Components of an expert system (adopted from David. (1994))*

ii. The Knowledge Base

This is a collection of facts and rules. The knowledge base is created from information provided by human experts. The knowledge base contains the domain knowledge useful for problem solving. In a rule-based expert system, the knowledge is represented as a set of rules. Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the IF (condition) and THEN (action) structure. When the condition part of a rule is satisfied, the action is executed.

iii. The inference engine

This carries out the reasoning whereby the expert system reaches a solution. It links the rules given in the knowledge base with the facts provided in the database. The inference engine is a generic control mechanism for navigating through and manipulating knowledge and deduce results in an organized manner. The non-expert user queries the expert system. This is done by asking a question, or by answering questions asked by the expert system. The inference engine uses the query to search the knowledge base and then provides an answer or some advice to the user.

2.1.2 Expert-Systems By De Kock (2003)

Intelligent - software systems are a specific implementation of Experts systems. Expert-systems are computer programs that contain knowledge and analytical skills of one or more human domain experts (Tripathi 2011). The main goal of expert-systems is to capture knowledge of a human expert then encode it into a software program in a way that the knowledge and expertise are available even in the absence of a human expert.

Expertise transferred from expert to computer involves a number of steps, among them knowledge acquisition, knowledge representation, knowledge inference and then knowledge presentation to the user. Knowledge is acquired from experts and documents then represented as rules known as the knowledge base, then there is a reasoning mechanism called the inference engine that manipulates rules from the knowledge base and draws data from the working memory to make conclusions (De Kock 2003). Figure 2.2 shows the components of an Expert-system and how they interact.

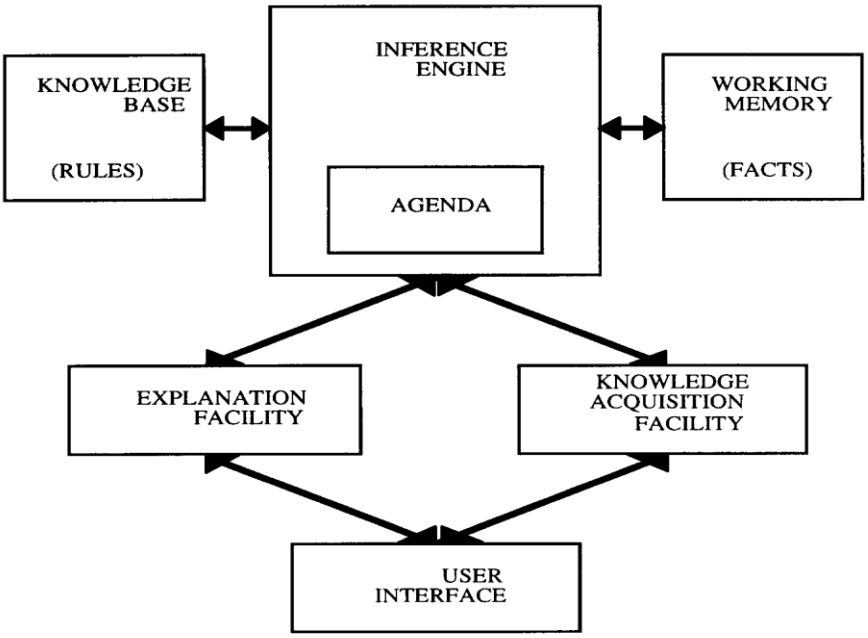


***Figure 2.2**: Expert-system (De Kock 2003)*

i. The User-interface

It is the boundary between the user and the system. Types of interfaces include question and answer, menu driven, natural language and graphic user-interface. The user provides input (text, voice or action) and receives expert explanations which are usually text from the knowledge base through the user-interface (De Kock 2003).

ii. The Knowledge Base

The knowledge base contains the knowledge necessary for understanding, formulating and for solving problems Gaševic, Djuric and Devedžic (2006). This component is made up of the data collected from human experts and documents of the domain of interest, the designer of a specific expert-system chooses how to represent the knowledge, and this may be through production rules, frames, logic, semantic nets or other methods (Tripathi 2011).

iii. The Inference Engine

This is the reasoning component of the expert-system, it forms the brain of the software, it is encoded with the rule interpreter to process and interpret rules. Its main task is to go through the large pool of rules in the knowledge base and then make reasonable conclusions which are then given to the user through the interface.

The traditional implementation uses the IF then ELSE production rules in which the inference engine acts as an interpreter that matches the conditions in the production rules with data instantiations in the working memory. The fuzzy implementation uses fuzzy production rules which use logic sets to characterise the variables and terms used in the propositions of rules, they are in the form IF antecedent THEN consequent, where the antecedent is in the form “x is Z” the truth value of this propositions is determined by the degree of matching between the linguistic variable “x” and the linguistic term “Z”. Another implementation that uses fuzzy production rules is the fuzzy inference system, which is used to model input and output relationships. Basically it is accomplished in four steps: 1) compare input with the membership functions in the antecedent; 2) compose the membership values to obtain the degree of support of each rule; 3) then generate a consequence of each rule depending on the degree of support and 4) resolve the output fuzzy set (Valdez et al. 2007).

iv. Explanation Facility

This component is one of the most important features of expert-systems. It is a subsystem that justifies the system`s actions. It helps to answer the “what” and “how” questions posed by the user, by explaining every decision that is made by the system. It thus serves as a tutor in sharing the system`s knowledge with the user (Tripathi 2011). Usually, the developer of an expert-system formulates a list of questions that might be asked by users on certain decisions taken by the system, the developer then provides answers which are then kept in a database. When a user asks for justification of a certain decision the system then searches from the list of pre-compiled questions then if a match is found it provides the response as justification.

v. Knowledge Acquisition Component

This component gives experts the capability to load knowledge into the knowledge base. Knowledge acquisition is the process of accumulation, transfer and transformation of problem solving expertise (De Kock 2003; Holder 2006). The implementation of this component differs from system to system, but it usually comprises of an input facility (e.g. web forms) that have rules to guide the content author through the process of curriculum-writing. The knowledge acquisition component simplifies the curriculum-writing process by restricting the user and guiding them, it then does the rest of the structuring in the background, and this means that content authors are spared the technical implementations of the system at hand (Valdez et al. 2007; Sowa 2000).

vi. The Working Memory

The working memory is a global database of facts used by the rules. It is the memory that contains information needed to make decisions while the expert-system is operating. The working memory represents knowledge or facts that are known so far. These facts are tested by the antecedent conditions of rules. This is achieved by comparing them with the facts in the knowledge base then coming up with the consequent.

The unbounded rule-based expert system for software development methodology expert systems is the implementations of expert-systems therefore, they follow the same ES architectural design. Even though, they add specific components they all tap from the same basic architecture of Expert-systems

2.1.3 Rule-Based Expert System

Loggia and Basili (1986) conducted an extensive research on expert systems and came up with the idea that expert systems can be described in two ways, based on the type of the interface namely:

i. Rule-based deduction

In this instance of inference, the domain knowledge base is represented as a set of rules in the form of if-then-else statements.

ii. Frame-based abduction

Information is represented in descriptive “frames” of information; it uses hypothesis-test cycles. When problem features are given, the system generates a set of potential causes/hypotheses which are then tested, and more questions are raised. This method mimics human reasoning. This study is proposing a system based on Rule-based Deduction instead of Frame-based abduction.

2.1.4 Human and Rules Following.

Do human experts follow rules like a computer program? This is a question that remains unanswered. Most of researchers have tried to answer this question in various ways. (Hartley, 1985) distinguishes between rule-following rules as:

i. Rule-following behaviour 1(RF1) ➡ this is the most general form. In this instance, the expert follows a set of predefined rules (follows a rulebook). The implementer does not need to be an expert but has to be good at following rules. For example, anyone can cook whatever dish they like if given a cookbook. Expertise is essential when rules fail.

ii. Rule-following behaviour 2(RF2) ➡ This technique is similar to RF1 though the rules are not external i.e. the expert learns the rules through continuous implementation and some rules are passed from expert to expert through communication.

iii. Rule-following behaviour 3(RF3) ➡ this is another technique which assumes that the rules are embedded in the subconscious mind. An example is language learning by a child, each language has rules, and children are able to follow these rules without knowing them. This instance also implies that rules can be followed by trial and error without actually being monitored or following a rulebook.

Since the researcher wants to develop a software agent, the only feasible rule-following behaviour is RF1 that will be used to develop the Unbounded Rule-based expert system. The computer always needs instructions to act on and a set of predefined rules (Rulebook) to control program execution (Hartley, 1985).

2.2 Software Process Models

Table 2.1: Software Process models

Models	Definitions
(A) Waterfall Model	The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is rigid and linear. Waterfall development has distinct goals for each phase of development where each phase is completed for the next one is started and there is no turning back. The perceived advantages of the waterfall process are that it allows for departmentalization and managerial control. A schedule is typically set with deadlines for each stage of development and a product can proceed through the development process. In theory, this process leads to the project being delivered on time because each phase has been planned in detail.
(B) Incremental development model	Various methods are acceptable for combining linear and iterative systems development methodologies, with the primary objective of each being to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process. There are three main variants of incremental development-namely: A series of mini-Waterfalls are performed, where all phases of the Waterfall are completed for a small part of a system, before proceeding to the next increment. Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system. The initial software concept, requirements analysis, and design of architecture and system core are defined via Waterfall, followed by incremental implementation, which culminates in installing the final version.

(C) Reuse-oriented model	The reuse-oriented model, also called reuse-oriented development (ROD), is a method of software development in which a program is refined by producing a sequence of prototypes called models, each of which is automatically derived from the preceding one according to a sequence of defined rules. However, these models can be subdivided to form a more specific model, including Spiral, XP, and Scrum, among other development methodologies.

According to Sommerville (2011), software process models are "a set of related activities that lead to the production of a software product". Some of software process models are shown in Table 2.1.

According to Fitzgerald et al. (2002), from time to time methodology is used interchangeably with method, however, Avison and Fitzgerald (2008:569) argued that the term methodology" is not similar to method as it has certain characteristics that are not implied by method", for example, the methodology includes the context of philosophy in which the user's opinions are taken into consideration. Eventually, it was concluded that methodology is a broader concept than method. Hence, Fitzgerald et al. (2002:5) use the term „formalized method" to refer to those commercial, brand-name and the in-house methods which are formally documented.

To aid in the understanding of various views of SDMs, a definition of terms from different researches was presented in Table 2.2.

Table 2.2: Definitions of Systems Development Methodologies

Author	**Definition of SDM**	**Key term**
Fitzgerald (2008:568	"A systems development methodology is a recommended means to achieve the development, or part of the development, of information systems based on a set of rationales and an underlying philosophy that supports, justifies and makes coherent such a recommendation for a particular context. The recommended means usually include the identification of phases, procedures, tasks, rules, techniques, guidelines, documentation and tools. They might also include recommendations concerning the management and organization of the approach and identification and training of the participants	Recognizes phase procedures, tasks, rules, techniques, guidelines, documentation and tools
Huisman and Iivari (2006:32)	Systems development methodologies may be described as the approach that constitute the four fundamental components; philosophical approach, phases, process model, and tools and techniques	Embraces philosophical approach, phases, process model, and tools and techniques

2.2.1 Components of Systems Development Methodology

The systems development method may be described as the approach that embraces four fundamental components namely: Philosophical approach, Phases, Process Model, and Tools and techniques (Huisman & Iivari, 2006).

i. Philosophical approach

Philosophical approach involves people's assumptions upon which the methodology is built, and characterizes development approaches such as structured, object-oriented and information modelling (Huisman & Iivari, 2006:32).

ii. Methodology

This includes the levels or steps needing to be carried out as a sequential process. It is a systematic way that is established on the basis of a particular philosophy guided by the emergent system (Avison & Fitzgerald, 2008:34). Method constitutes a set of guiding principles, beliefs and values, set of goals and system processes that force interpretations and actions, and examples include: OMT, IE, etc. (Huisman & Iivari, 2006).

iii. Process Model

This is the logical analysis that describes what an emerged system wants to achieve and how it is going to achieve it (Avison & Fitzgerald, 2008). As a consequence, it reflects the sequence of steps in the course of systems development. Some of the examples are linear life-cycle and spiral models (Huisman & Iivari, 2006).

iv. Tools and Techniques

Techniques help to ensure thorough decision-making in finding the necessary design in the development of information systems. Examples include DFD, ERD, etc. Therefore, tools are perceived to be mechanisms aimed at helping to develop information systems e.g. software packages (Avison & Fitzgerald, 2008).

In summary, it is evident from the aforementioned definitions that SDM is one of the fundamental issues of information systems to be applied during the development. It is commonly defined as an attempt to attend to complex and difficult activities in trying to compose the emerged information system. Furthermore, SDMs are defined as the

components aimed at rationalising and effecting better control in the process of the development of information systems.

2.2.2 The use and effectiveness of SDMs in organizations

The use and effectiveness of systems development methodologies will be discussed through the following concepts namely: usage, adoption, in-house versus commercial methods, managers versus developers and non-use of SDMs.

a) The use of systems development methodologies

Although the methodological practices have been popularized in the academic research domain, businesses have not been able to universally embrace them (Griffin & Brandyberry, 2008). It is worth pointing out that the use of systems development methodology is not compulsory in organizations, hence practitioners are not persuasive in employing them during the development of IT projects (Pieterse, 2006). As a result, the adoption of systems development methodologies by practitioners has been somewhat slow (Laudon, 2000).

In practice, not many software development organisations use SDMs, the common perception from previous studies being that many software developers are reluctant, for some reason, to use or adopt systems development methodologies (SDMs) in the course of developing IT projects (Laudon, 2000:388). The opposite is true with the study by McLeod (2007), where it shows that the measure of systems development methodology usage is more advanced than previous empirical studies might suggest

In most cases, choices about the adoption of the standard methodology must be made by the IS department. However, the business managers should be the ones to make such decisions, as they are the ones to make investigations in terms of money, effort, time and ultimately business decisions (Avison & Fitzgerald, 2008:574). In view of that, organizations adopting the methodology are spending enormously as the cost includes training of staff and users, additional hardware and software, ongoing consultancy cost that might rise up to the level of the initial cost of the actual methodology (Avison & Fitzgerald, 2008).

In a similar vein, Fitzgerald et al. (2002:97) believe the choice that organizations are making with regard to the usage of the methodology in a given project depends on pragmatic reasons rather than the fundamental features of the method itself. This leads to the widespread observation that methodology users to a greater extent, have different perspectives in the usage or adoption of systems development methodologies (Huisman & Iivari, 2006:29).

The research conducted by Pieterse (2006) indicated that organizations adopt the systems development methodology due to the following reasons:

i. A financial increase is acquired
ii. There is a deficiency in knowledge
iii. There are time limitations
iv. Methodologies are not popularly applicable as yet.

In addition, Saini et al. (2009) outlined the following reasons advanced to justify the use of systematic development methodologies:

i. They make it easier to understand systems by making a separation between conceptual design, logical design, and physical design
ii. They design systems so that they are extendable and easily maintained
iii. They move toward automated design tools and dynamic page generation
iv. They are easier to use in order to manage development in the form of costs, time, task allocation, deliverables, etc.
v. They overcome reliance on designers who constructed the system
vi. They are used to reduce risks associated with shortcuts and mistakes
vii. They are used to produce documentation that is consistent from one project to the next

Systems development projects are not homogenous by nature; there is no single-best one-size-fits-all methodology; and in practice, most organizations are not committed to any single methodology they mix and match as appropriate.

b) The adoption of systems development methodologies

Though the choice of methodologies made by organizations depends on the particular needs they experience, yet to some organizations the methodology adopted still needs to be amplified by writing the detailed manual according to the requirements of their development staff (Avison & Fitzgerald, 2008).

In real development practice, formalized ISD methods are mostly not used in their entirety, nor as they were initially aimed for by their creators, even though they offer the template to guide the development practices. This brings into play the notion of method-in-action which is inferred to be uniquely enacted by the developer to meet any development project goal (Fitzgerald et al., 2002).

In contrast, it is tempting to assume that one methodology can be universal, in that a single methodology can be applied to all kinds of projects throughout the organisation. Linked with an inappropriate recognition of developer-specific factors, IS development methodologies have always been deemed to become the „one size fits all" solution to the IT project in the organisations (Vidgen et al., 2002:4).

To rectify this misconception, more than one systems development methodology may probably be used to accomplish a project goal in the organization; in which, practically speaking, a methodology could be adopted in a particular phase of a project (Yahya et al., 2002:25). However, the same methodology will not be interpreted and applied in the same way by different developers; neither will the same developer apply the same systems development methodology in the same way in more than one emergent project or in different development situations (Fitzgerald et al., 2002:13). Generally, systems development methodologies have different levels of complexity and rigour (Griffin & Brandyberry, 2008). Consequently, this triggers the decision as to whether to use in-house or commercial methods which are discussed in the subsequent section.

c) In-house versus commercial methods

Regarding commercial SDMs and in-house methodology usage, it has been determined that most often organizations use in-house methods as an alternative, and these methods may be adapted from some form of commercial method (Akmanligil & Palvia, 2004:45; Masrek et al., 2008:144; McLeod, 2007:567). The reason for this is that the in-house methodologies are perceived to be a better fit for the developmental context requirement's need and are more flexible than commercial methodologies (Kiely & Fitzgerald, 2003:10; Fitzgerald et al., 2002:166).

Of those organizations using a formal methodology, many use methodologies by engaging in contracts with consulting companies; whereas others develop their own in-house methodologies to suit their needs, sometimes using other methodologies as a template or guide. Certainly, the choice as to whether any one or a specific methodology would be used will be determined by the condition of buyouts or mergers (Griffin & Brandyberry, 2008:2).

d) Managers versus developers

Systems development methodologies reflect management's agenda in the sense that IS managers are more pessimistic regarding the support and positive effect that systems development methodologies provide than it seems with systems developers (Huisman & Iivari,

2006:41). Seemingly, the perceptions of IS managers of systems development methodologies are more positive than those of the developers (Huisman & Iivari, 2006). Another point to consider is that with high-power distance, top executives or senior partners may exercise power to determine the way tasks should be carried out (e.g., possibly contrary to any IS development methods, or adjust them to suit any given situations). Conversely, with low-power distance, less authorized employees may feel empowered by the use of systems development methods, because they are less likely to be overruled (Fitzgerald et al., 2002:128). Typically, Fitzgerald et al. (2002:128) hold the contentious view that commercial methods are more likely to be useful to inexperienced developers as the standard methods in-house are often favoured by experienced developers.

e) Reasons for not using systems development methodologies

While there are a number of significant arguments in the favour of systems development methodologies, there are also a number of arguments and pressures that question the use of such methodologies. Fowler's (2001) contentious issue is that, although the methodologies have existed for a long time, they have not been notable for being terribly successful; and little has been known about them. Moreover, methodologies are often heavy and finely detailed, and this can appear to obstruct the productivity (Griffin & Brandyberry, 2008:2). As a result, the nature of ISDMs (i.e. difficult, too abstract, too many deliverables to be generated) prevents them from been widely used in the organizations, hence, it is vital to look at criteria of the methodology prior to attempting to use it. Clearly, for a methodology to be feasibly applied, it should be easy to understand, and be appropriate to be used in any types of cases (complex, big or small systems) and also be relevant across the development life cycle (Yahya et al., 2002).

In summary, it is apparent from the aforesaid realities that the use of SDMs has always been considered as capable of improving the quality and productivity of systems development (Yahya et al., 2002:15; Masrek et al., 2008:143). Thus, the efficiency of the development team and the quality of the developed product are increased. Nevertheless, there is little empirical evidence that can prove these assumptions as the systems are still overwhelmed by challenges of cost-overrun, being time-consuming, and failure to meet the users" requirements (Fitzgerald et al., 2002:1).

2.2.3 The advantages and benefits of systems development methodologies

In this sub-section, the advantages and benefits of systems development methodologies will be discussed through the following aspects: better end-product, better development,

standardized process and procedures, supporting tool, knowledge and education, communication, systematic guide and benefit.

2.2.4 Better end-product

Systems development methodologies (SDMs) are perceived to be an attempt to improve the quality of the end-product of the information systems development process (Mihailescu & Mihailescu, 2009; Griffin & Brandyberry, 2008:2; Huisman & Iivari, 2006:33; Yahya et al., 2002:15; Masrek et al., 2008:143). As a result, systems development methodologies are aimed at meeting the system's requirements, completely within the budget and the schedule (Avison & Fitzgerald, 2008:570; Yahya et al., 2002:15; Saini et al., 2009:89; McLeod, 2007:563).

According to Avison and Fitzgerald (2008:570), people have the need to have better information systems; therefore, they use systems development methodology in an attempt to improve the end product of the development process. The components of the quality of an information system include acceptability, availability, cohesiveness, compatibility, documentation, ease of learning, effectiveness, efficiency, fast development rate, flexibility, functionality, implement ability, low coupling, amenability, portability, reliability, robustness, security, simplicity, testability, timeliness and visibility. The maximization of this criteria could potentially lead to a better product during the development of Information systems (Avison & Fitzgerald, 2008:570).

2.2.5 Better development

According to Yahya et al. (2002:26), systems development methodologies are also helpful in the development of an information system in that they involve certain types of tools such as CASE tools, word-processing, spreadsheets, graphic tools, presentation software and charting tools. Thus, these tools tend to ease the project development. Similarly, information systems development methodologies bring together various, often vendor-specific procedures, techniques, tools and documentation aids relevant to different sections of the information systems development life-cycle (Nandhakumar & Avison, 1999:176; Mihailescu & Mihailescu, 2009:2).

Avison and Fitzgerald (2008:570) state that there are outputs (deliverables) likely to accumulate in every phase of a systems development methodology during information systems development. It is assumed that since the emergence of systems development methodologies has become apparent, the levels of skills required of professionals have been

reduced, and observably, the costs are expected to be reduced while the development of information systems through the implementation of systems development methodologies would be improved.

2.2.6 Standardizing processes and procedures

Typically, methodologies entail the standard processes that aid project managers in achieving the goals of the project. Nonetheless, the use of systems development methodologies differs from one organization to another; hence, it is vital to discern how organizations feel about the adoption of a systems development methodology (Yahya et al., 2002:15).

During the development of a system, a standard and common process is followed, and this makes it easy to integrate systems. Thus, staff members become familiar with the process and this accrues to the benefit of staff members in gaining skills and knowledge and consequently makes it easy for them to change from project to project without being retrained (Avison & Fitzgerald, 2008:570).

According to Futrel et al. (2002:107), a standard for IT systems of the Federal Republic of Germany comprises reasons for the importance of a standardized process. The purpose of these processes is to achieve the following objectives:

- The advancement and the assurance of the quality;
- The successful results to be delivered might be guaranteed by the standardized procedures; and
- The cost for the implementation of methodology can be easily checked.

It is these standard procedures that enable the costs calculation to be viable and visible and aid to recognize any risks that could emerge, in that they minimize the use of resources and reduce friction losses among stakeholders; as a result, progress of the project is liable to be effectively monitored (Futrel et al., 2002:107).

The other point to make is that standard procedures reduce misunderstandings among all stakeholders, as the requirements and the needs are effectively communicated and agreed-upon. As a result, all parties involved will be efficiently supported through their underlying needs (Futrel et al., 2002:107).

Pieterse (2006: IV) came up with the idea that the developers are of the opinion that a systems development methodology produces a system of high quality. Therefore, it proves that the systems development methodology has major strengths such as consistency and well-being

structuring. Hence, it relies on certain sets of procedures and steps to be followed when implemented.

2.2.7 Supporting tool

A systems development methodology is deemed to be one of the most essential supporting tools to achieve complicated practices that comply with the preface of information systems development (Yahya et al., 2002:26). Hence, the role of the SDM is assumed to be supporting the potential stakeholder in meeting their requirements and helps managing or solving the problem during the systems development (Mihailescu & Mihailescu, 2009:3).

2.2.8 Knowledge and education

Certainly, SDMs serve as an instrument to transfer knowledge between experienced and novice developers and templates to guide the development practice of newly emerged systems (Fitzgerald, 2002:120). Thus, methodology in use also serves as the framework for the knowledge-work processes that underpin the business IS application with which the team is involved (Meso et al., 2006:15), hence education and training are adhered to as the benefit in systems development methodologies (Huisman & Iivari, 2006:41). As a consequence, as an individual unit of adoption, ISDM is perceived to be a potential means which, if used, improves job performance within the business. The deployment of ISDM is perceived to improve the career of individual developers and cater for support or transfer of knowledge base within the organization or community (Mihailescu & Mihailescu, 2009:4).

2.2.9 Communication

Furthermore, systems development methodologies allow feasible communication among all stakeholders (Futrel et al., 2002:107; Mihailescu & Mihailescu, 2009:5). Effectively, the use of systems development methodologies improves the communication between developers and users. (McLeod, 2007:568). In particular, Extreme programming (XP) is another type of methodology which underpins the communication and coordination among the team members to a greater extent; hence it promotes the teamwork spirit among the stakeholders (Qumer & Henderson-Sellers, 2008).

2.2.10 Systematic guide

In practice, many consulting companies have developed and implemented various commercial systems development methodologies that offer systematic guidance during the development of information systems (Leem & Kim, 2002:249; Meso et al., 2006:17). Notably, standard

methods offer detailed guidelines that inform the action of developers and the specifications for the tasks required to be carried out during the systems development (McLeod, 2007:563).

Certainly, one of the major reasons for IT projects to fail to meet the requirements is because project managers operate reactively rather than proactively. When a problem appears, they are hesitant to make effective and viable decisions, which often results in a situation that could destroy a project. It is assumed that this behaviour occurs due to a lack of guidelines to aid in the development process. In view of that, practices of systems development methodology are believed to serve as a guide necessary for the development of the project (Avison & Fitzgerald, 2008:574), and as a result methodology is recommended to deliver proper and required functions, on time, within budget, and that meet user expectations (Wetherbe, 2001:604).

2.2.11 Benefit of using SDMs

According to Saini et al. (2009), the use of systematic development methodologies can be of great value to companies in that:

i. It reduces cycle time and / or reduced maintenance;
ii. It increases productivity;
iii. There are higher quality systems; and
iv. It improves project management; time and cost estimates, tracking and resource utilization.

In the study by Masrek et al. (2008:143), it was found through empirical evidence that the use of systems development methodologies provides benefits in the development of information systems. The following benefits are arranged from the highest ranking to the lowest ranking: The systems development methodology was perceived to:

i. Allow better project control;
ii. Facilitate communication among developers;
iii. Ensure that systems meet user needs better;
iv. Increase visibility of systems development process;
v. Improve the quality of the systems developed;
vi. Help ensure that documentation is produced;
vii. Facilitate interchangeability of developers among projects;
viii. Increase the likelihood that systems will be delivered on time;
ix. Reduce maintenance of systems; and
x. Increase the likelihood that systems will be delivered within budget.

Moreover, the survey by Yahya et al. (2002:15) discovered the following benefits when carrying out information systems development projects in the adoption of methodologies.

i. Higher quality of produced documents;
ii. Increases in the involvement of the computer user;
iii. Increases in the number of information systems products;
iv. A reduction of the maintenance cost;
v. A reduction of the design error;
vi. Fulfilment of the user's requirements; and
vii. An improvement of communication between the user and the systems developer.

The significant feature of systems development methodologies is that they provide a reminder that triggers knowledge and act as guides to interpretation and action in IS development projects, thus promoting communication, coordination, control, and production (Mihailescu & Mihailescu, 2009:5). With such conceptualization, the claim that SDMs are aimed at improving the development of information systems in terms of productivity and quality becomes apparent, nevertheless, there is little evidence to prove the veracity of such a remark (Yahya et al., 2002:15). As a consequence, this controversy invokes the proposal of the next section in which the disadvantages and critics of systems development methodologies are discussed.

2.3 Disadvantages and criticisms of systems development methodologies

In this sub-section, disadvantages and criticism of systems development methodologies will be discussed with reference to their being time-consuming, with complex development, a need for method tailoring and not being widely practised as well as lacking full control.

2.3.1 Time-consuming during development

According to Truex et al. (2000:54), though systems development methodologies for developing information systems have always been reported as being significant in the information systems discipline, there are still problems associated with their practicality. The main problems are that systems take time to develop, there is an overrun of cost, and the initial requirements for the system are not met.

In the early days, as envisaged, SDMs were designed to address the complexities and difficulties during the process of developing information systems. Nevertheless, the inherent problem was that the system still failed to be developed on time, within the budget, or be of good quality (Fitzgerald et al., 2002). In view of that, Wetherbe et al. (2001:629) state that methodology application often takes time to complete, hence they operate well in large

projects with clearly-defined requirements where there is no pressure to complete the project quickly. According to the study conducted by Pieterse (2006: IV), the results indicated that the use of systems development methodology in organizations is time-consuming, and as a result, low-quality systems are taken out in order to gain a faster delivery time.

2.3.2 Complex development

Methodologies are often considered to be process-heavy, and are likely to obstruct productivity due to their finely detailed requirements (Griffin & Brandyberry, 2002). Due to such step-by-step techniques and the fact that every step must be completed before the next one can commerce, the development may be delayed. Hence, organizations are still hesitant to apply systems development methodologies during information systems development (Laudon, 2000:388). This complexity results in the entire development process not being fully successfully accomplished as per purported objectives (McLeod, 2007:569).

In effect, the study by Yahya et al. (2002:31) ascertains that even though the methodology is used in organizations, because of the perception attached that they improve productivity, quality and communication; yet they are not uniformly accepted due to their complexity (i.e. difficult, too abstract, too many deliverables to be generated). Practically, inconsistency makes the process analysis of methodologies unreliable, thus making it difficult to reach the process improvements (Dahiya & Jain, 2010).

Furthermore, Dahiya and Jain (2010) assume that it is of the utmost importance to follow the processes defined in the methodologies in a consistent and accurate manner across all projects, thus ensuring the effective use of methodologies. However, these become challenging tasks due to:

i. Software-development processes being complex, with their step-by-step techniques of interdependent activities (Griffin & Brandyberry, 2002; Dahiya & Jain, 2010). Most of the processes and methodologies bring about reference documents only. As a result, it makes it difficult for practitioners to gain knowledge about the processes from the document references and follow them in their projects (Dahiya & Jain, 2010).
ii. Practitioners follow the processes manually and deliver the necessary data into various life-cycle tools that do not have much integration with the engineering tools. These different sets of tools hardly allow an accurate methodology implementation across multiple projects, and this generates data and reports inconsistent with the organization (Dahiya & Jain, 2010).

2.3.3 Need for method tailoring

It is apparent from the aforementioned statements that SDMs do not fully satisfy organizational requirements and they are not suitable to every development project even if they are justified particularly in the areas of productivity, quality and communication (Yahya et al., 2002:27; Fitzgerald et al., 2002:3). In view of that, methods are not perceived as relevant to practice, they are just being utilized because they appear plausible, hence the method-in-action (myriad of contextual factors pertaining to the process of development) has a great effect on the outcome of the development (Fitzgerald et al., 2002:xii). Practically, systems development methodologies are frequently not fit for the development situation, and this can lead to the development of a method tailoring strategy to create the methodin-action to meet a particular situation (Fitzgerald et al., 2002:100).

In such contexts, even with the notion of systems development methodologies, the underlying problems in the systems development have not been eliminated. One among other reasons for the failure has been that practitioners do not realize the benefits and needs of systems development methodologies; hence they are not in shape with the complexity of the development situations (Fitzgerald et al., 2002:3).

2.3.4 Not widely practised

Systems development methodologies usage has been highly advertised in the academic research arena for many years, but it has not been universally embraced by business even if IT projects" failure is often experienced in the organizations (Griffin & Brandyberry, 2008:2; Mihailescu & Mihailescu, 2009:1). With such conceptualization, the evaluation of the reality of SDMs has not been keeping pace with their growth; therefore, the veracity of their usefulness and effectiveness has not been accurately understood in the organizations (Siau & Tan, 2005).

2.3.5 Lack of the human factor

Methodologies such as Structured Systems Analysis and Design Method (SSADM) have been claimed to be playing a crucial role to the success of systems development, however, they fail to note the human, social, and organizational aspects; instead, they emphasize the technical aspects of IS development. The deployment is not undertaken with people in mind, and this leads to users not being fully satisfied with the successful implementation of the system (Yardley, 2002:83). Depressingly, Jackson Systems Development (JSD) was aimed at minimizing the responsibility of the developer in the development process; thus, creating the illusion that the ability and insight of the developer are confined to the systems development methodology (Fitzgerald et al., 2002).

2.3.6 Lack of full control

Another point to make is that some of the practices of systems development methodologies are carried out by external companies that specialize in providing information systems development services, it is alleged that the companies may lose control over the functions of their information systems and instead depend on technical influence and the breakthrough of external vendors (Laudon, 2000:384).

2.4 Software development methodologies

According to Scientiae, Science and Campus (2013), a software development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. One system development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations. Centers for Medicare and Medicaid Services (CMS) has considered each of the major prescribed methodologies in context with CMS' business, applications, organization, and technical environments. As a result, CMS requires the use of any of the following linear and iterative methodologies for CMS systems development, as appropriate. The following are the methodologies identified:

2.4.1 The Waterfall

With Waterfall model, project is divided into sequential phases as shown in Figure 2.3, with some overlap and splashback acceptable between phases. There is more emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.

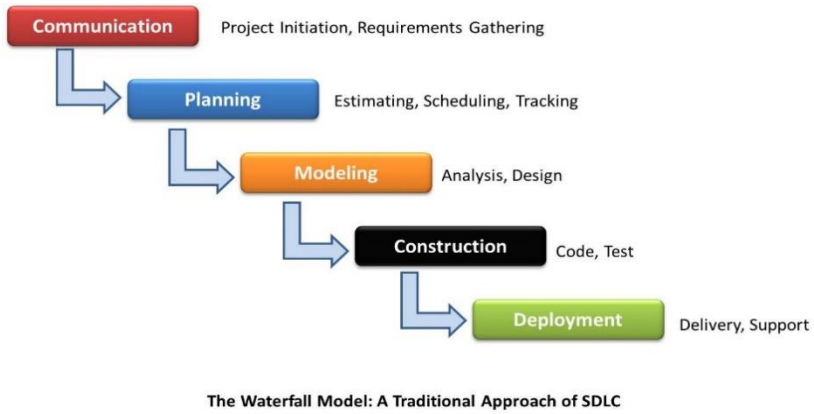


***Figure 2.3**: The Waterfall Model*

Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

The basic Principles of a waterfall model are as follows:

i. Project is divided into sequential phases, with some overlap and splashback acceptable between phases.
ii. Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
iii. Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

The strengths of this model are as follows:

i. Ideal for supporting less experienced project teams and project managers, or project teams whose composition fluctuates.
ii. The orderly sequence of development steps and strict controls for ensuring the adequacy of documentation and design reviews helps ensure the quality, reliability, and maintainability of the developed software.
iii. Progress of system development is measurable.
iv. Conserves resources.

The weaknesses of this model are as follows:

i. Inflexible, slow, costly and cumbersome due to significant structure and tight controls.
ii. Project progresses forward, with only slight movement backward.
iii. Little room for use of iteration, which can reduce manageability if used.
iv. Depends upon early identification and specification of requirements, yet users may not be able to clearly define what they need early in the project.
v. Requirements inconsistencies, missing system components, and unexpected development needs are often discovered during design and coding.
vi. Problems are often not discovered until system testing.

vii. System performance cannot be tested until the system is almost fully coded, and under-capacity may be difficult to correct.
viii. Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged.
ix. Produces excessive documentation and keeping it updated as the project progresses is time-consuming.
x. Written specifications are often difficult for users to read and thoroughly appreciate.
xi. Promotes the gap between users and developers with clear division of responsibility.

Situations where most appropriate to use the waterfall:

i. Project is for development of a mainframe-based or transaction-oriented batch system.
ii. Project is large, expensive, and complicated.
iii. Project has clear objectives and solution.
iv. Pressure does not exist for immediate implementation.
v. Project requirements can be stated unambiguously and comprehensively.
vi. Project requirements are stable or unchanging during the system development life cycle.
vii. User community is fully knowledgeable in the business and application.
viii. Team members may be inexperienced.
ix. Team composition is unstable and expected to fluctuate.
x. Project manager may not be fully experienced.
xi. Resources need to be conserved.
xii. Strict requirement exists for formal approvals at designated milestones.

Situations where least appropriate to use waterfall:

i. Large projects where the requirements are not well understood or are changing for any reasons such as external changes, changing expectations, budget changes or rapidly changing technology.
ii. Web Information Systems (WIS) primarily due to the pressure of implementing a WIS project quickly; the continual evolution of the project requirements; the need for experienced, flexible team members drawn from multiple disciplines; and the inability to make assumptions regarding the users' knowledge level.
iii. Real-time systems.
iv. Event-driven systems.
v. Leading-edge applications.

2.4.2 Prototyping

The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete as shown in Figure 2.4. The basic Principles of prototype are as follows:

i. It is not a standalone, complete development methodology, but rather an approach to handling selected portions of a larger, more traditional development methodology (i.e., Incremental, Spiral, or Rapid Application Development (RAD)).

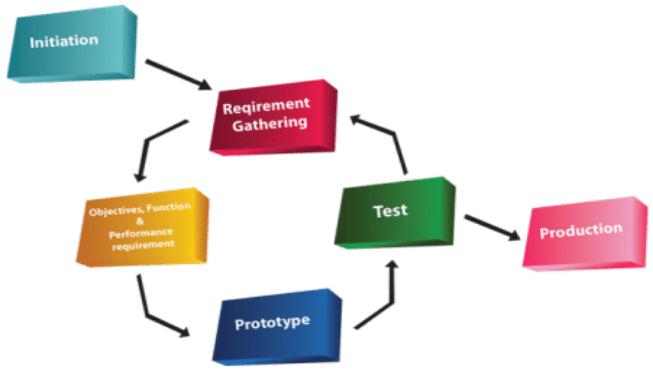


***Figure 2.4**: Prototyping*

ii. Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
iii. User is involved throughout the process, which increases the likelihood of user acceptance of the final implementation.
iv. Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.
v. While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.
vi. A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.

The strengths of this model are as follows:

i. The model addresses the inability of many users to specify their information needs, and the difficulty of systems analysts to understand the user's environment, by

providing the user with a tentative system for experimental purposes at the earliest possible time (Janson and Smith, 1985).

ii. Can be used to realistically model important aspects of a system during each phase of the traditional life cycle (Huffaker, 1986).

iii. Improves both user participation in system development and communication among project stakeholders.

iv. Especially useful for resolving unclear objectives; developing and validating user requirements; experimenting with or comparing various design solutions; or investigating both performance and the human computer interface.

v. Potential exists for exploiting knowledge gained in an early iteration as later iteration are developed.

vi. Helps to easily identify confusing or difficult functions and missing functionality.

vii. May generate specifications for a production application.

viii. Encourages innovation and flexible designs.

ix. Provides quick implementation of an incomplete, but functional, application.

The weaknesses of this model are as follows:

i. The approval process and control is not strict.

ii. Incomplete or inadequate problem analysis may occur whereby only the most obvious and superficial needs will be addressed, resulting in current inefficient practices being easily built into the new system.

iii. Requirements may frequently change significantly.

iv. Identification of non-functional elements is difficult to document.

v. Designers may prototype too quickly, without sufficient up-front user needs analysis, resulting in an inflexible design with narrow focus that limits future system potential.

vi. Designers may neglect documentation, resulting in insufficient justification for the final product and inadequate records for the future.

vii. Can lead to poorly designed systems. Unskilled designers may substitute prototyping for sound design, which can lead to a “quick and dirty system” without global consideration of the integration of all other components. While initial software development is often built to be a “throwaway”, attempting to retroactively produce a solid system design can sometimes be problematic.

viii. Can lead to false expectations, where the customer mistakenly believes that the system is “finished” when in fact it is not; the system looks good and has adequate user interfaces but is not truly functional.

ix. Iterations add to project budgets and schedules; thus, the added costs must be weighed against the potential benefits. Very small projects may not be able to justify the added time and money, while only the high-risk portions of very large, complex projects may gain benefit from prototyping.
x. Prototype may not have sufficient checks and balances incorporated.

Situations where most appropriate to use prototype model:

i. Project is for development of an online system requiring extensive user dialog, or for a less well-defined expert and decision support system.
ii. Project is large with many users, interrelationships, and functions, where project risk relating to requirements definition needs to be reduced.
iii. Project objectives are unclear.
iv. Pressure exists for immediate implementation of something.
v. Functional requirements may change frequently and significantly.
vi. User is not fully knowledgeable.
vii. Team members are experienced (particularly if the prototype is not a throw-away).
viii. Team composition is stable.
ix. Project manager is experienced.
x. No need exists to absolutely minimize resource consumption.
xi. No strict requirement exists for approvals at designated milestones.
xii. Analysts/users appreciate the business problems involved, before they begin the project.
xiii. Innovative, flexible designs that will accommodate future changes are not critical.

Situations where least appropriate to use prototyping model:

i. On a mainframe-based or transaction-oriented batch systems.
ii. Web-enabled e-business systems.
iii. Project team composition is unstable.
iv. Future scalability of design is critical.
v. Project objectives are very clear; project risk regarding requirements definition is low.

2.4.3 The Incremental

Combination Linear and Iterative framework: The incremental build model is a method of software development where the product is designed, implemented and tested incrementally.

A little more is added each time until the product is finished. Figure 2.5 shows the phases of incremental model.

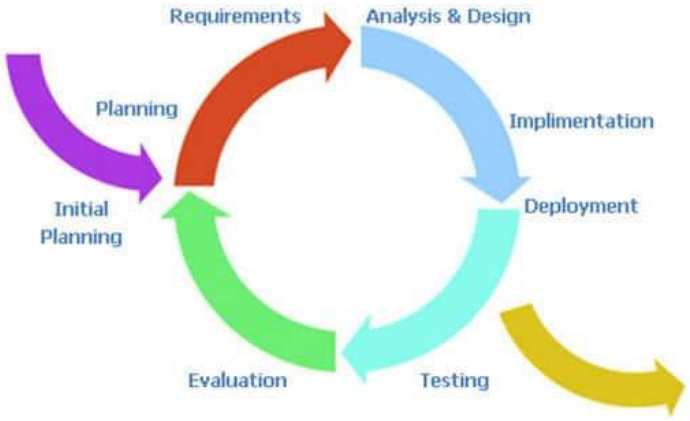


***Figure 2.5**: The incremental model*

The basic Principles of incremental model are as follows:

i. A series of mini-Waterfalls are performed, where all phases of the Waterfall development model are completed for a small part of the system, before proceeding to the next increment; OR
ii. Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of the system, OR
iii. The initial software concept, requirements analysis, and design of architecture and system core are defined using the Waterfall approach, followed by iterative Prototyping, which culminates in installation of the final prototype (i.e., working system).
iv. Various methods are acceptable for combining linear and iterative system development methodologies, with the primary objective of each being to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process:

The strengths of this model are as follows:

i. Potential exists for exploiting knowledge gained in an early increment as later increments are developed.
ii. Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval/signoff by the user and information technology management at designated major milestones.

iii. Stakeholders can be given concrete evidence of project status throughout the life cycle.
iv. Helps to mitigate integration and architectural risks earlier in the project.
v. Allows delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.
vi. Gradual implementation provides the ability to monitor the effect of incremental changes, isolate issues and make adjustments before the organization is negatively impacted.

The weaknesses of this model are as follows:

i. Since some modules will be completed much earlier than others, well-defined interfaces are required.
ii. Difficult problems tend to be pushed to the future to demonstrate early success to management.
iii. When utilizing a series of mini-Waterfalls for a small part of the system before moving on to the next increment, there is usually a lack of overall consideration of the business problem and technical requirements for the overall system.

Situations where most appropriate to use this model are:

i. Large projects where requirements are not well understood or are changing due to external changes, changing expectations, budget changes or rapidly changing technology.
ii. Web Information Systems (WIS) and event-driven systems.
iii. Leading-edge applications.

Situations where least appropriate this model are:

i. Very small projects of very short duration.
ii. Integration and architectural risks are very low.
iii. Highly interactive applications where the data for the project already exists (completely or in part), and the project largely comprises analysis or reporting of the data.

2.4.4 The Spiral

The spiral model is a risk-driven software development process model. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or

more process models, such as incremental, waterfall, or evolutionary prototyping. Figure 2.6 shows the model and the phases involved.

***Figure 2.6**: The Spiral model*

The basic Principles of the spiral model are as follows:

i. This model focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle.
ii. “Each cycle involves a progression through the same sequence of steps, for each portion of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program.” (Boehm, 1986).
iii. Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration (Boehm, 1986 and 1988).
iv. Begin each cycle with an identification of stakeholders and their win conditions and end each cycle with review and commitment (Boehm, 2000).

The strengths of this model are as follows:

i. The model enhances risk avoidance.

ii. Useful in helping to select the best methodology to follow for development of a given software iteration, based on project risk.
iii. Can incorporate Waterfall, Prototyping, and Incremental methodologies as special cases in the framework, and provide guidance as to which combination of these models best fit a given software iteration, based upon the type of project risk. For example, a project with low risk of not meeting user requirements, but high risk of missing budget or schedule targets would essentially follow a linear Waterfall approach for a given software iteration. Conversely, if the risk factors were reversed, the Spiral methodology could yield an iterative Prototyping approach.

The weaknesses of this model are as follows:

i. It is highly customized to each project, and thus is quite complex, limiting reusability.
ii. A skilled and experienced project manager is required to determine how to apply it to any given project.
iii. There are no established controls for moving from one cycle to another cycle. Without controls, each cycle may generate more work for the next cycle.
iv. There are no firm deadlines. Cycles continue with no clear termination condition, so there is an inherent risk of not meeting budget or schedule.
v. Possibility exists that project ends up implemented following a Waterfall framework.

Situations where most appropriate to use this model:

i. Real-time or safety-critical systems.
ii. Risk avoidance is a high priority.
iii. Minimizing resource consumption is not an absolute priority.
iv. Project manager is highly skilled and experienced.
v. Requirement exists for strong approval and documentation control.
vi. Project might benefit from a mix of other development methodologies.
vii. A high degree of accuracy is essential.
viii. Implementation has priority over functionality, which can be added in later versions.

Situations where least appropriate to use this model are as follow:

i. Risk avoidance is a low priority.
ii. A high degree of accuracy is not essential.
iii. Functionality has priority over implementation.
iv. Minimizing resource consumption is an absolute priority.

2.4.5 Rapid Application Development (RAD)

Rapid application development is a form of agile software development methodology. Unlike Waterfall methods, RAD emphasizes working software and user feedback over strict planning and requirements recording. Figure 2.7 shows how the model works and the process involved.

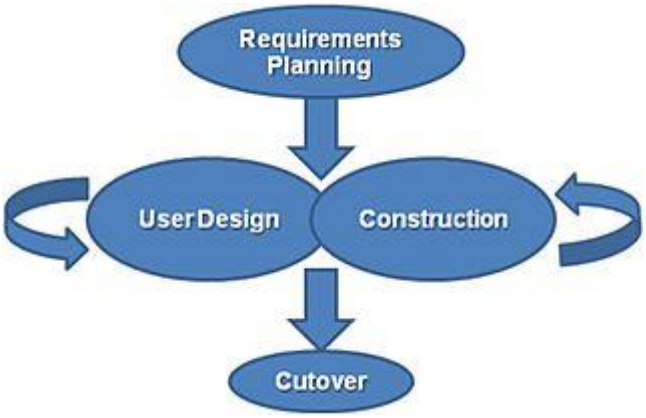


***Figure 2.7**: Rapid Application Development (RAD)*

The basic Principles of this model are as follows:

i. Key objective is for fast development and delivery of a high-quality system at a relatively low investment cost.
ii. Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
iii. Aims to produce high quality systems quickly, primarily through the use of iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. These tools may include Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), fourth-generation programming languages, code generators, and object-oriented techniques.
iv. Key emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.
v. Project control involves prioritizing development and defining delivery deadlines or time boxes. If the project starts to slip, emphasis is on reducing requirements to fit the time box, not in increasing the deadline.
vi. Generally, includes Joint Application Development (JAD), where users are intensely involved in system design, either through consensus building in structured workshops, or through electronically facilitated interaction.
vii. Active user involvement is imperative.

viii. Iteratively produces production software, as opposed to a throwaway prototype.
ix. Produces documentation necessary to facilitate future development and maintenance.
x. Standard systems analysis and design techniques can be fitted into this framework.

The strengths of this model are as follows:

i. The operational version of an application is available much earlier than with Waterfall, Incremental, or Spiral frameworks.
ii. Because RAD produces systems more quickly and to a business focus, this approach tends to produce systems at a lower cost.
iii. Engenders a greater level of commitment from stakeholders, both business and technical, than Waterfall, Incremental, or Spiral frameworks. Users are seen as gaining more of a sense of ownership of a system, while developers are seen as gaining more satisfaction from producing successful systems quickly.
iv. Concentrates on essential system elements from user viewpoint.
v. Provides the ability to rapidly change system design as demanded by users.
vi. Produces a tighter fit between user requirements and system specifications.
vii. Generally, produces a dramatic savings in time, money, and human effort.

The Weaknesses of this model are as follows:

i. More speed and lower cost may lead to lower overall system quality.
ii. Danger of misalignment of developed system with the business due to missing information.
iii. Project may end up with more requirements than needed (gold-plating).
iv. Potential for feature creep where more and more features are added to the system over the course of development.
v. Potential for inconsistent designs within and across systems.
vi. Potential for violation of programming standards related to inconsistent naming conventions and inconsistent documentation.
vii. Difficulty with module reuse for future systems.
viii. Potential for designed system to lack scalability.
ix. Potential for lack of attention to later system administration needs built into system.
x. High cost of commitment on the part of key user personnel.
xi. Formal reviews and audits are more difficult to implement than for a complete system.

xii. Tendency for difficult problems to be pushed to the future to demonstrate early success to management.
xiii. Since some modules will be completed much earlier than others, well-defined interfaces are required.

Situations where most appropriate to this model:

i. Project is of small-to-medium scale and of short duration (no more than 6 man-years of development effort).
ii. Project scope is focused, such that the business objectives are well defined and narrow.
iii. Application is highly interactive, has a clearly defined user group, and is not computationally complex.
iv. Functionality of the system is clearly visible at the user interface.
v. Users possess detailed knowledge of the application area.
vi. Senior management commitment exists to ensure end-user involvement.
vii. Requirements of the system are unknown or uncertain.
viii. It is not possible to define requirements accurately ahead of time because the situation is new or the system being employed is highly innovative.
ix. Team members are skilled both socially and in terms of business.
x. Team composition is stable; continuity of core development team can be maintained.
xi. Effective project control is definitely available.
xii. Developers are skilled in the use of advanced tools.
xiii. Data for the project already exists (completely or in part), and the project largely comprises analysis or reporting of the data.
xiv. Technical architecture is clearly defined.
xv. Key technical components are in place and tested.
xvi. Technical requirements (e.g., response times, throughput, database sizes, etc.) are reasonable and well within the capabilities of the technology being used. Targeted performance should be less than 70% of the published limits of the technology.
xvii. Development team is empowered to make design decisions on a day-to-day basis without the need for consultation with their superiors, and decisions can be made by a small number of people who are available and preferably co-located.

Situations where least appropriate to use this model:

i. Very large, infrastructure projects; particularly large, distributed information systems such as corporate-wide databases.

ii. Real-time or safety-critical systems.
iii. Computationally complex systems, where complex and voluminous data must be analysed, designed, and created within the scope of the project.
iv. Project scope is broad and the business objectives are obscure.
v. Applications in which the functional requirements have to be fully specified before any programs are written.
vi. Many people must be involved in the decisions on the project, and the decision makers are not available on a timely basis or they are geographically dispersed.
vii. The project team is large or there are multiple teams whose work needs to be coordinated.
viii. When user resource and/or commitment is lacking.
ix. There is no project champion at the required level to make things happen.
x. Many new technologies are to be introduced within the scope of the project, or the technical architecture is unclear and much of the technology will be used for the first time within the project.
xi. Technical requirements (e.g., response times, throughput, database sizes, etc.) are tight for the equipment that is to be used.

2.5 Factors Influencing Choice of An Adequate Software Development Methodology

Geambaşu (2011) and Al Ahmar (2010) concur that the following are the factors that influence the selection of the Software Development Methodology:

a) Project time

Projects that have short time schedules are well suited for methodologies that are designed to increase the speed of development. Prototyping, iterative development and XP are excellent choices when project time is short because they best enable the project team to adjust the system functionality on the basis of a specific delivery date, and if the project schedule starts to slip, it can be readjusted by removing functionality from the version or prototype under development. The waterfall methodology is the worst choice when time is critical because it does not allow for easy schedule changes.

b) Clarity of user requirements

When the user requirements are unclear or subject to change, it is difficult to understand them by talking about them and explaining them with written reports. Users normally need to interact with the software to really understand what the new system can do. System prototyping,

throwaway prototyping, and XP are usually more appropriate when user requirements are unclear or unstable because they provide prototypes for users to interact with early in the SDLC.

c) Familiarity with technology

When the system will use new technology with which the system analysts and programmers are not familiar, early application of the new technology in the SDLC will improve the chance of success. If the system is designed without some familiarity with the technology, risks increase because the tools may not be capable of doing what is required. Throwaway prototyping is particularly appropriate for a lack of familiarity with technology because it explicitly encourages the developers to develop design prototypes for areas with high risks. Iterative development is good as well because it creates opportunities to investigate the technology in some depth before the design is complete. Although one might think system prototyping would also be appropriate, it is much less so, because the early prototypes that are built do not investigate the new technology deeply. Usually, it is only after several prototypes and several months that the developers discover problems in the new technology.

d) System complexity

Complex systems require careful and detailed analysis and design. Throwaway prototyping is particularly well suited to such detailed analysis and design, but system prototyping is not. The traditional structured methodologies can handle complex systems, but without the ability to get the system or prototypes into users' hands early on, therefore, some key issues may be overlooked.

e) Schedule visibility

Determining whether a project is on schedule is one of the challenges in Software systems development. Methodologies in which design and implementation occur at the end of the project are 'poor' regarding this criterion (e.g. waterfall development) whereas methodologies that move many of the critical design decisions to an earlier point in the project can help project managers recognize and address risk factors and determine whether a project is on schedule.

2.5.1 Geambasu Additional Factors

Geambaşu et al., (2011) identified the following factors that influence the decision of choosing a software development methodology, clarity of the initial requirements, accurate initial estimation of costs and development time, incorporation of requirements changes during the

development process, obtaining functional versions of the system during the development process, software criticality, development costs, length of the delivery time of the final system, system complexity, communication between customers and developers, size of the development team.

Those factors apply on Rational Unified Process (RUP), Rapid Application Development (RAD), and Extreme programming (XP). The following are the factors identified.

a) Clarity of the initial requirements

Rational Unified Process. Correct and complete definition of the requirements from the beginning of the project represents the starting point for the development of software that incorporates all the functionality required by the client.

Rapid application development methodology is composed of a variable number of prototyping cycles and consists in building, step by step, viable software. There are held several iterative sessions. The full functional requirements are not set at the beginning of the project but are specified in detail by the users in each iterative session.

Extreme programming. In the Exploration phase of XP methodology, the development team members meet with the clients at a planning meeting. The clients define the requirements of the software as “user-stories”. It is not necessary that the initial requirements fully describe the functionalities of the final system because the methodology is composed of multiple short development cycles and the requirements are updated in each development cycle.

b) Accurate initial estimation of costs and development time

Rational Unified Process. In the first phase of RUP methodology – Inception, is defined the project scope, are identified the risks, is chosen a strategy to mitigate the identified risks, is drawn up an initial model use cases based on the defined requirements and are planned the activities that will be performed during the whole development process. All these elements lead to a realistic initial estimation of costs and development time of project.

Rapid application development (RAD). It is a type of incremental model. In RAD model, the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. For each of the five phases components of RAD methodology is established a maximum number of days for accomplishing the objectives of the phase. Depending on the specifics of each project, the effective development time can be estimated with a small margin of error. The

initially estimated development costs are subject to change depending on the effort involved by the implementation of the requirements that are changed during the development process.

Extreme programming (XP). It is very difficult to estimate the effort required for the development of the entire system because not all the requirements are known at the beginning of the project. Extreme Programg differs from traditional methodologies primarily in placing a higher value on adaptability than on predictability. Proponents of XP regard ongoing changes to requirements as an often natural and often inescapable aspect of software development projects; they believe that being able to adapt to changing requirements at any point during the project life is a more realistic and better approach than attempting to define all requirements at the beginning of a project and then expending effort to control changes to the requirements.

XP prescribes a set of day-to-day practices for managers and developers; the practices are meant to embody and encourage particular values. Proponents believe that the exercise of these practices which are traditional software engineering practices taken to so-called "extreme" levels leads to a development process that is more responsive to customer needs ("agile") than traditional methods, while creating software of similar or better quality.

c) Incorporation of requirements changes during the development process

Rational Unified Process. The subdivision of phases in iteration allows the developers to make the necessary changes to adapt to new requirements during the whole development process, but the cost of change, especially in the late stages of development, is high. The aspects regarding the management of the changes made during the development process are specified in the Configuration and Change Management discipline in RUP methodology

Rapid application development. The information system is divided into smaller segments, which facilitates making changes along the development process, at any time in the cycle.

Extreme programming. The methodology is flexible and can easily adapt to changes in the requirements. System changes can be made even in late stages of the life cycle for its adaptation to customer requirements.

d) Obtaining functional versions of the system during the development process

Rational Unified Process. Iterative development of a system leads to multiple versions of the system throughout its life cycle, versions that must be carefully managed to avoid the integration, at the end of the development process, of incomplete solutions.

Rapid application development (RAD). The methodology allows users to interact with variants of the system from early stages due to the use of prototypes.

Extreme programming (XP). Working versions of the developed system are frequently obtained. This way of working helps customers understand the progress in the development of the system and allows him to stop the project after a number of completed iteration if he does not have enough available funds.

e) Software criticality

Rational Unified Process. The methodology comprises a discipline – Project Management that aims at tracking the proper running of the development process by managing the risks from the initiation of the project and by adequately planning and monitoring the iterations in order to achieve project objectives. Risks management involves identification of the risks and elaboration of strategies for their mitigation. Test discipline has as purpose to ensure the proper functioning of the system. There are used techniques to check and validate the proper implementation of the defined and designed requirements and there are identified the situations that may cause disruptions. The methodology can be used with success for the development of software with a high level of criticality.

Rapid application development (RAD). During the development process, there are numerous tests conducted, but as the team is working quickly through project iterations, it is possible to miss information and requirements and the system quality may be lower than in the case of using a traditional approach, as RUP.

Extreme programming (XP). Checking the proper functioning of the developed software is achieved by using two types of tests: "unit test" and "functional test". “Unit tests” are written by developers before adding a new functionality to the system and then run continuously. In this way are checked parts of code (classes, methods, etc.). "Functional test" are specified by the customer and usually refers to checking the functioning of the whole system. Although there are run tests for checking the proper functioning of the software, there still is a risk related to quality assurance because XP methodology does not have a structured review process to reduce the deficiencies.

f) Development costs

Rational Unified Process. The high complexity of the methodology requires the use of a large number of resources, including financial.

Rapid application development (RAD). Due to reusability of the prototypes and to the short development time, the methodology can be used in conditions of relatively low costs.

Extreme programming (XP). The methodology is organized in a number of short development cycles, thus trying to reduce the cost of changes made to adapt to the requirements expressed by the customers during the system life cycle. The development team is small which implies low costs of human resources.

g) Length of the delivery time of the final system

Rational Unified Process. Software development using RUP methodology involves conducting a large number of activities to meet project objectives, leading to extended delivery time of the final system. The methodology can be adapted to fit specific requirements of a particular project, but the adaptation process is also complex.

Rapid application development. RAD methodology focuses on the limitation of time, being defined delivery deadlines (time-boxes) for project components. If there are problems with meeting the deadlines, the focus is on reducing requirements, rather than on increasing deadlines. Therefore, we can say that the objective of RAD methodology is to deliver the minimum set of requirements necessary in the shortest time period.

Extreme programming (XP). The methodology gives importance to customer satisfaction, by delivering software when is needed and not in a distant future, when the requirements might already be changed.

h) System complexity

Rational Unified Process. The decision to use RUP methodology for software development should take into account the technical and project management complexity. It is recommended to use this methodology when the complexity of both factors is above average.

Rapid application development. The methodology can be used with success in case of small or medium projects where the scope is well defined. RAD tends to fail when used for the development of complex systems or of distributed systems.

Extreme programming (XP). The methodology emphasis communication between client and developers, thus XP is not recommended for large projects because is difficult to maintain the communication with a large group. Also, the use of XP is problematic in case of complex

projects with many interdependencies. XP should be used when the system complexity is medium or low.

i) Communication between customers and developers

Rational Unified Process. The customers provide to the developers, information on the requirements of the future system and give them, when required, a feedback on certain results of the development process, but they are not actively involved through the whole process of software development.

Rapid application development. There is a direct participation of customers in the process of software development. The customers participate to working sessions, along with the developers, being actively involved in the process of defining requirements, as well as in the process of evaluating and validating the prototypes and the final software.

Extreme programming. One of XP principles refers to On-site customer. This means that a representative of the future users of the system must be available throughout the development process to answer the questions of the development team.

j) Size of the development team

Rational Unified Process: The high complexity of the methodology requires the use of a large number of human resources.

Rapid application development.: This methodology is not recommended in case of large project teams or when there are many people required to make decisions. It works best with small or medium projects.

XP: The development team should be small, between 2 and 10 people.

2.6 Rules for SDM selection

Studies conducted by Zaeid, (2013) and Al Ahmar (2010) reveal that many authors have written about rules that govern the selection of a software development methodology (SDM). According to Shrivastava et al. (2010), these rules as defined in the rulebook of an expert system are:

i. be specialist: known facts and procedural rules
ii. use heuristics: interpolate from known facts

iii. Justify its conclusions: to establish credibility and confidence. The user can ask: how do you know a particular fact? Why do you ask a particular question?
iv. be able to learn: be able to absorb new knowledge and apply it
v. estimate the reliability of its answer

A research done by Centers for Medicare and Medicaid Services (2008) reveals that each methodology has its own strengths and weaknesses. These weaknesses and strengths can be used in formulating selection criteria by considering the properties of a project and comparing those using features of each SDM, one can make a choice.

We will utilize the features in section 2.6 to tackle the problem of selecting the most appropriate software development methodology. If each methodology is represented as an object, then its list of situations where desirable define the state it is possible to come up with a Rulebook that will be used as a reference by the inference engine to compare the user`s inputs of project properties to the available data then a correct decision made on which model better suits the project at hand.

2.7 Expert systems for automaton of Methodology process

The literature review shows that many of expert systems that automate the methodology selection system have been constructed (Al Ahmar, 2010). Some expert systems prototypes are briefly described in the next subsections.

2.7.1 Software Development Methodologies Expert System

Zaeid et al. (2013) described how a prototype can be created with the use of Software Development Methodology Expert System. He proposed a rule based expert system which consists of three phases namely:

i. <u>The identification phase</u>

The processes involved in the identification phase are:

i. Technical problem situation;
ii. Social problem situation;
iii. Socio-technical problem situation; and
iv. Complex problem situation

ii. <u>The determination phase:</u>

In this phase, a group of methodologies are selected based on the clarity of user requirements, development time, and familiarity with technology and system criticality as shown in Table 2.3.

Table 2.3: Problem Situations and Corresponding Methodologies (Adopted from Zaied et al. (2013)

Problem Situation	Methodologies
Technical problem situation	Traditional methodologies, Agile methodologies, Rapid Application Development methodologies
Social problem situation	Holistic methodologies
Socio-technical problem situation	Socio-Technical methodologies
Complex problem situation	Framework

iii. The Selection Phase

This is phase at which a specific methodology is selected based on the objectives. The prototype has the following components:

i. Knowledge acquisition subsystem
ii. Knowledge base
iii. Inference engine
iv. Explanation subsystem
v. User interface

The above specification will produce a very good agent, but it would not be too efficient due to the number of screens for the selection to be completed. These simply means that the user will navigate through many screens before getting to the desired output. This might prove to be a frustrating procedure. Given that users do not have an in-depth knowledge of the problem domain, are likely to make mistakes if asked many questions. Thus, much input might lead to erroneous output.

Zaied et al. (2013) proposed a prototype that uses selection criteria which include:

i. Clarity of user requirements.
ii. Project time
iii. Complexity of system
iv. System reliability
v. Schedule visibility

vi. Familiarity with technology

Extreme Programming was used as an example by Zaied (2013) on the proposed prototype.

Zaied, (2013) defined Extreme Programming (XP) is a software engineering methodology, the most prominent of several agile software development methodologies. Like other agile methodologies, Extreme Programming differs from traditional methodologies primarily in placing a higher value on adaptability than on predictability. An example of XP is shown in Table 2.4.

Table 2.4: Examples of XP (Adopted from Zaeid et al., 2013)

Project property	XP
With short time schedule	Excellent
Unclear user requirements	Excellent
Unfamiliar with technology	Poor
Complex system	Poor
Reliable system	Good

Zaeid (2013) adopted the two stages of inference from Al Ahmar (2010) which are:

1. Uses the **if then else** statement to assign point values to each SDM e.g.

   If: user requirements clarity is low

   Then: check SDMs property and assign then 0; 0.5 or 1.0 points for poor, good or excellent respectively.

2. The selected SDMs are then ranked.

If a particular SDM has at least one 0.0 value in any selected criteria, it is not displayed. Points are added for the selected criteria then they are displayed in descending order with the top most being the most desirable.

2.7.2 Al Ahmar Software Development Methodologies Expert system

Al Ahmar (2010) developed his Software Development Methodologies-Expert System (SDMES) using the same inference engine from Figure 2.8 shows the main screen of SDMES where various options are displayed. Al Ahmar (2010) asserts that the button titled View Available SDMs allows the user to browse and read all present data on the currently available SDMs. The user can start a consultation session by clicking on the button titled Consultation.

The user can also enter the user manual and get more help by selecting the HELP button or exit the system by clicking Exit. Other major screens and features of the user interface will be described by the following sample consultation: Consider a software project that is mainly characterized by the following key challenges: It is a complex software system, with unclear user requirements, and limited project time (i.e., short time schedule).

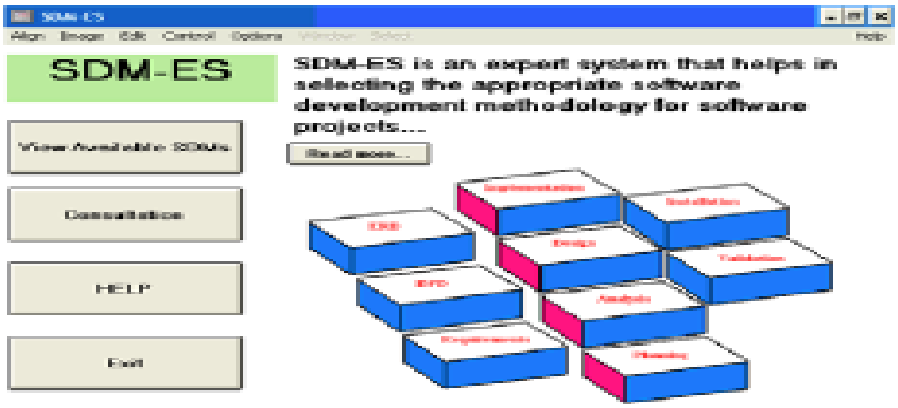


***Figure 2.8**: Components of an expert system (adopted from: Al Ahmar (2010)*

2.7.3 Short coming of Zaeid and Al Ahmar prototype models

i. The short comings of Zaeid et al. (2013) are
ii. Limited selection criteria – There are many criteria that can be used to select a methodology thus it is not appropriate to limit the developer to only six.
iii. Also, the knowledge used is static, meaning that the data is loaded during construction, and if any changes are to be made, then the developer must be summoned.

The shortcomings of Al Ahmar (2010) model are:

i. Poor interface – It is not easy to navigate through the prototype.
ii. Too many forms – it takes time to get solution.

2.7.4 Common features in the reviewed systems

After reviewing the identified two expert system s development methodology by Al Ahmah (2010) and Zaeid (2013), the researcher found some components that are present in all the systems. However, these components are named differently although they implement the

same functionalities. The relationship between components of an expert system and the ones in SDMES show that some components are basically built using the general architecture of expert systems. However, Software Development Methodologies Expert System implement their own unique features like the Expert Model which acts as a consultant component.

2.7.5 Previous similar studies.

Table 2.5 presents the outcomes of previous studies regarding the structuring of SDMs expert system, usage and effectiveness of systems development methodologies (SDMs) in a specific IT project.

Table 2.5: Previous studies about SDMs

No	Findings	Author
1	Expert-systems are computer programs that contain knowledge and analytical skills of one or more human domain experts	Tripathi 2011
3	Expert System (ES) is a computer program which is designed to carry out tasks associated with a human expert. It is a program which tries to do things which are usually regarded as the province of human, involving judgments and decision-making	David, (1994).
3	Rule-Based Expert System	Loggia & Basili (1986)

Table 2.5: continued

4	The study shows that more than 50% of companies developing software employ some systems development methodologies	Griffin and Brandyberry (2008:8)
5	Factors influencing choice of an adequate software development methodology	Geambaşu (2011) and Al Ahmar (2010)
66	From the results obtained, 61.1% had agreed that the use of SDMs improved the information systems development; in that it could be completed on schedule or on time	Yahya *et al.* (2002:26)

7	The research suggests that competencies and SDMs deployed in a given development situation are intertwined in such a way that they cannot be separated in practice	Omland and Nielsen (2009:222)
8	It was recognized that RUP methodology together with PRINCE 2 methodology complemented each other and allowed any project management team to improve the control of the entire process and allow for more efficient change management.	Lancaster University (2005:25)
9	It is suggested that the main purpose of SDMs is to advance human capabilities which would allow human agents to perform valuable functions. Thus, lack of attention to all these three components, i.e. structure, agency, and cultural system, will reduce the success of SDMs intervention.	Mihailescu and Mihailescu (2009:6)
10	SDMs are thought of as a guide to organizations for purposes of the achievement of the vision rather than a prescriptive basis for project planning and action.	Madsen *et al.* (2006:237)
11	It was found that the perceptions of IS managers of systems development methodologies were more positive than those of developers.	Huisman and Iivari (2006:40)

Table 2.5: continued

12	Methods seem more like idealizations than prescriptions, and might better be presented as "cases" or "exemplars" instead of practical frameworks.	Truex *et al.* (2000: 74)
13	The study suggests that effectively matching SDMs to an application domain does leverage knowledge management outcomes by influencing the effectiveness of knowledge-work processes in a systems development project.	Meso *et al.* (2006: 26)
14	Methods are no more applied blindly to the development process to ensure the success of project development process; instead they are tailored to fit the particular development project.	Kiely and Fitzgerald (2005:12)
15	The study found that an organization is probably unwise to use heavily prescriptive SDMs to improve its software development performance.	Middleton and McCollum (2001:18)
16	It is determined that organizational culture is a factor affecting successful adoption of an agile method	Strode *et al.* (2009: 7)
13	It was observed that levels of methods and process formalization constitute one of the contextual factors within company infrastructure categories that influenced software development practices.	Bern *et al.* (2007:8)
14	The findings have indicated that the organization use most of an in-house development SDMs in their systems development projects.	Masrek *et al.* (2008:144)
15	Practitioner experience suggests that agile methods are particularly suitable for projects where requirements are more abstract and difficult to define; thus, such organisations have either not adopted or moved away from traditional approaches	Toleman *et al.* (2004:469)

2.8 Hardware platforms

There are different hardware platforms that each software needs to run on, such as the web, mobile, and desktop platform. Web applications have been gaining popularity due to accessibility, and efficient use of resources. These applications run on web servers and their interfaces are usually presented in HTML and accessed through web browsers. The other

platform is the desktop applications and are defined as stand-alone applications that run on laptops and desktop computers. These applications do not rely on the web but on desktop application (Computer Basics: Understanding Applications, 2014).

Most recently, the Mobile platform has become more famous than any other. The mobile applications are small pieces of software that run on mobile devices (Lee, 2013). The researcher choses mobile platform to run his proposed SDMES because this platform is likely to achieves the goals of this research than the other platforms.

Mobile applications are considered to be better than web applications due to ubiquity of smartphones, manifoldness of possibilities; the ubiquity of application stores; unmatched user experience; proximity to customers and better visibility. In addition, these distribution platforms are stores, and unlike the web or its search engines, stores are designed to sell and present products (Jahns, 2010).

2.9 Software platforms

For all the mobile platforms, the proposed system will run on android due to various reasons, such as the fame of android devices and their penetration rate in Africa. Also, by making it an android application will ensure that it reaches a larger population. The percentage of android users in Africa compared to other platforms such as iPhone Operating System (IOS) is higher. Statistics show that there are more android users than other mobile platforms (Rollins & Sandberg, 2013). Android applications are eased to develop and deploy. Android is hailed as the first complete, open, and free mobile platform, for the designers took a comprehensive approach when they developed this platform. Androids have secure operating systems and built a robust software framework and they also allow for rich application development opportunities.

Android applications unlike, other operating systems like iPhone Operating System and blackberry, are open-sourced, and the tools needed for development of androids applications are free and there is no need to purchase special developer devices as a developer there is also no registration needed for your hardware (Durga, 2014). There are no licensing or royalty fees to develop on the platform, no membership, certification and testing fees requirements. Android applications can be easily distributed and commercialized in a variety of ways (“Introducing Android,” 2009).

2.9.1 Development Platform tools

Android is a Linux-based operating system that was designed so that its applications are open-sourced and so are the development tools. Below is a brief description of most famous android development tools.

2.9.1.1 Android software development kit (SDK)

Android software development kit is a development package that provides the developer with a set of application programming interface (API) libraries and developer tools necessary to build, test and debug applications. Developers can easily download the abstract data type (ADT) bundle to quickly start development. This package includes the Android software development kit components and a version of eclipse integrated development environment (IDE) with built-in Android developer tools, Android platform-tools, latest android platform and the latest Android system images for the emulator (10 Hot Consumer Trends 2014, 2014). Android software development kit will be used in this study.

2.9.1.2 Adobe Air

This is another package that enables developers to package the same code into native applications for different mobile operating systems (“Adobe Air,” 2014). The possibility of developing cross platform mobile applications is due to the fact that it runs on Action Script, Hypertext Mark-up Language (Html), and Cascading Style Sheets (CSS) AIR(“ADOBE FLEX 4.6 and ADOBE FLASH BUILDER 4.6,” 2011). An advantage of this development package is its ability to produce cross-platform applications. This essentially means an increase in the target population. It also saves development time and costs as the developer only develops once unlike other tools, whereby the developer has to develop separate applications for each mobile operating system.

2.9.1.3 Native Development Kit (NDK)

This is a development package that contains a set of tools which allow android developers to implement parts of their applications using native code languages such as C++. One advantage of using this development package is that developers can effectively reuse existing code from libraries written in these native languages (Qian, Zhu & Li, 2012). However, allowing flexibility for the developer by using a language that they are comfortable with, does not help much if it negatively affects the performance of applications. Developers need to balance such gains and drawbacks as the native code does not always increase performance but generally increases complexity (“10 Hot Consumer Trends 2014,” 2014). Thus, developers are urged to

only use NDK if it is essential to the application and not because they simply prefer to program in C++.

2.9.1.4 Titanium Mobile software development kit

The Titanium software development kit gives developers the ability to create quality native, mobile web or rich hybrid applications that can run on all platforms from a single codebase using JavaScript. It is an open source software development kit with a lot of community developers contributing constantly to improve it ("Titanium SDK," 2014). One would choose to use this software development kit because traditional approaches use native development tools and languages for each mobile operating system. This is a drawback because it requires management of multiple development projects and codebases. This is costly and time-consuming. If one wants to develop a hybrid application easily, it will be wise to use Titanium software development kit.

2.10 The Unbounded Rule-based expert system

The problem of selecting a suitable ISDM has been addressed in different ways by many researchers. McConnell S (1996) provided general guidelines on how to select the most suitable development lifecycle and gave practical tips on best practices for various development environments. Similarly, different authors of agile SDM, Highsmith A and Highsmith J (2002) provided general recommendations for adaptation and use of these SDM. Although these general guidelines and tips were very useful; they did not include recommendations for selection of a specific project at the hand. Cockburn A (2002) provided a decision model that helps select the suitable SDM from a family of ISDMs named Crystal. According to Vavpotič & Vasilecas (2012) organizations dealing with IS development often having lack knowledge and experience to objectively evaluate different types of SDMs. So the challenge is to automate the process of selecting SDMs and enable decision maker to select form different types of SDMs so that the professionals simply enter some information about the project and get an indication of the most appropriate methodology so the proposed URB-ES gives the flexibility to accommodate all the short coming from the previous studies.

The number of information systems development methodologies has increased and software engineers have struggled to select methodology appropriate for all applications. But there is no single methodology that will work for all development situations. Then the challenge is how to select the appropriate SDMs? Therefore, it is useful to use expert system as a selection tool to automate the process of selecting the suitable SDM. Expert systems are interactive computer programs that mimic and automate the decision making and reasoning processes

of human experts in solving a specific domain problem, through delivering expert advice, answering questions, and justifying their conclusions Zaid et Al (2013).

After identifying the gaps to rectify the shortcomings of the prototypes described in section 2.6, the proposed URB-RS will have a predefined knowledge base but will allow each manager to modify the knowledge base by adjusting the SDM models to suit their requirements or to add new SDMs to the knowledge base, and the URB-ES will let the user select the criteria they want to use without any bounds. The researcher will develop an application that runs on cheap and easily accessible devices. It will be a native application, thus ensuring that even those without proper internet access can use it, this comes as an advantage as it ensures mobility; users do not need to have Wi-Fi or data to use the application.

2.11 Summary

In this chapter, we reviewed literature on expert-systems and on specific implementations of the unbounded rule-base expert systems with the main intensions of discovering how they work and identify components of each of these reviewed systems. Reviewed documents show that most expert systems share the same design architectures. Although these are minor differences, all are derived from the general architecture of expert systems. This led to the conclusion that all unbounded rule-base expert systems are specialized implementations of the same base architecture; therefore, the blueprint of our proposed system is also derived from the general architecture of expert systems. For most unbounded rule-base expert systems, that we analysed the main domain knowledge representation is in the form of scripts. The disadvantage of this type of knowledge coding however that is it is mandatory for script writers to understand the internal workings of the system. This chapter also contains descriptions of platforms (Hardware and software). The discussion enabled an understanding of different platforms, identification of their advantages and disadvantages then the selection of the most appropriate for this study. Most of this information will be used as a base for the system requirements elicitation, for example, the strengths and weaknesses of reviewed systems. It also helped in making a decision on which hardware, software and development platforms to consider when designing and developing our prototype. In the next chapter will present the methodology of this study.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

A research methodology assists to guide the conception of the research and direct the investigation undertaken in this study. This chapter describes the research methodology followed, specifically a model by Oates (Oates, 2005) which is a description of some software development methodologies. This research followed the steps taken by Oates for his model in Figure 3.1.

3.2 Research design

The purpose of this study was to develop a software system; hence it involves the creation of a software artefact. The shaded blocks indicate the sequence of activities that were carried out in this course of study.

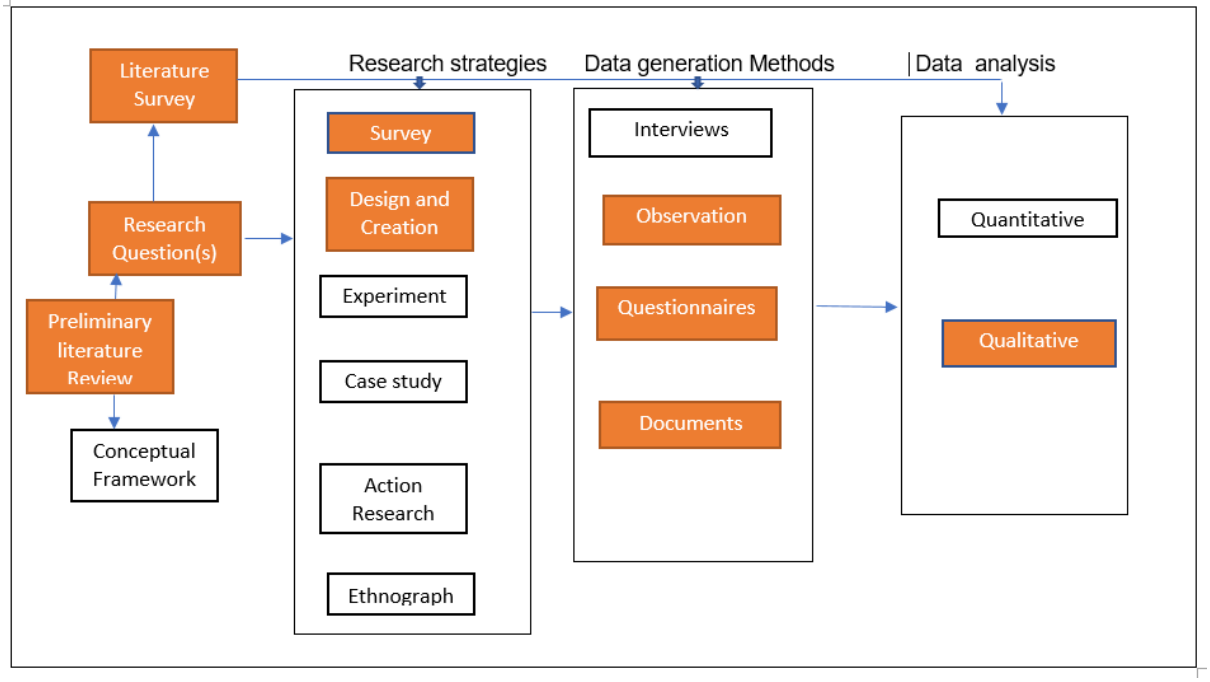


***Figure 3.1**: Model of research by (Oates, 2005)*

The research methodology used in this research is mixed method design but mostly inclined towards qualitative research since a greater part of the research needs an in-depth study of how human experts' reason. In the process of planning the research design of the URB expert system, the study needed to identify what tools are appropriate to develop an efficient, accessible and effective URB expert system. These comprise of the platforms for which the application will run, the operating system that will host the system as well as development tools.

3.3 Population of the study and Target population

The methodology adopted in this study proceeded in two phases: the first phases used a survey to investigate factors critical for the choice of a software development methodology. The survey was conducted amongst 101 project managers and software developers based at the iLAB (pty) Ltd Software Development Company based at Gauteng Sandton and a group of honours students based at the University of Johannesburg, South Africa. Both descriptive and inferential statistics were used to analyse the survey results. From the results of the analysis, the researchers then developed an architecture for an unbounded rule-based expert.

3.4. Data Sources and Data Collection Techniques

The following are the data sources and data collection techniques:

i. Observations

The study used structured observation method to collect data from the respondents (Wild & Diggines, 2015). The researcher observed participants making decisions on which methodology they will choose during the initiation phase of the software development project.

ii. Focus groups

Focus groups were used after the observation phase. The researcher led the informal open discussions where participants were discussing about how they were handling other software development projects, the reasons behind selecting the software development methodology and if they were any guidelines followed for choosing the methodology (Wild & Diggines, 2015).

iii. Literature review

The researcher read documents written about creation of intelligent systems; Read literature about human reasoning techniques and read documents about software development methodologies

iv. Questionnaires

Questionnaires were distributed to the target group of this study as designed at the Appendix.

3.5 Data analysis

The purpose of data analysis was to interpret and draw conclusions from the mass of collected data. Thematic Content Analysis (TCA) was used to analyse qualitative data Rosemarie (2007). During analysis, the researcher identified patterns or themes across the collected data (Braun and Clarke, 2006:29). Rosemarie (2007) stated that, “with TCA, the researcher groups and distils from the texts a list of common themes in order to give expression to the communality of voices across participants”. TCA often enables frequency counts and allows for quantitative analysis of initially qualitative data (Braun and Clarke, 2006:29). A number of software packages are available to automate the labelling and grouping of texts collected for qualitative researches and are especially useful in the analysis of numerous transcripts. Microsoft Word was used for labelling and grouping of texts in this research project. The Statistical Package for Social Sciences (SPSS) version 23.0 software was used to analyse data. Cross tabulations were done to analyse relationships between variables. Results emanating from the analysis were represented in the form of tables, graphs and charts. Further, inferential analysis based on Factor Analysis, Correlation, Regression Analysis was undertaken to aid in addressing the second objective focused on factors influencing the decision to adopt a software development methodology. Factor analysis was used as a dimension reduction technique given that the survey instrument had not been validated before. Correlation and Regression Analysis was used as explanatory techniques to address the second objective.

3.6 Development of the Prototype

After selecting the mobile platform as discussed in section 2.9, the next task was to choose between creating a native application or web-based application. Both web-based and native applications have their advantages and disadvantages. The native mobile application was to be chosen for this study. The researcher chose the native mobile application because web applications require a connection to function and have performance issues such as being slow in loading and unavailability, which may frustrate users. The native mobile application is not affected by the above-mentioned shortcomings. When users are on the go without Wi-Fi coverage, they have to pay for connection time to network operators. In some countries, the cost is minimal, but in others, the cost of data can be exorbitant (“Mobile Web Apps vs. Mobile Native Apps : How to Make the Right Choice,” n.d.). Native applications have better accessibility because they place a logo in the application list screen, providing visibility on a daily basis.

The researcher discovered that it was impossible to elicit all system requirements. There was need to expose a working model at an early phase in order to gather and refine requirements.

According to Somerville 2011, prototyping comes in handy in such situations. There was also a need to come up with design requirements, though the system architecture was borrowed from the generic architecture of expert-systems, there was need to adjust it to specific requirements, so prototypes were used to test various designs before the implementation phase. From the literature reviewed, the researcher chose the incremental development model to develop the proposed system because various methods are acceptable for combining linear and iterative systems development methodologies.

3.7 Ethical considerations

Ethical considerations were ensured throughout the study in order to protect the rights of the participants. The researcher sought permission to conduct the study, informed consent, voluntary participation, confidentiality, privacy and protection of participants from any harm were also considered.

3.7.1 Permission to conduct the study

The research proposal was presented at the School of Management Sciences and later to the University of Venda Higher Degrees Committee for quality assurance and approval. An application was made to the Research Ethics Committee of the University of Venda for ethical clearance to conduct the study. After getting the ethical clearance from the Ethics Committee, then the researcher started collecting data.

3.7.2 Informed consent

Before obtaining written consent forms, respondents were provided with information regarding the purpose and objectives of the study, the voluntary nature of participation and the time it will take to fill out the questionnaire. After being given the choice to choose whether to take part in the study, only those willing to participate were given the consent forms to complete.

3.7.3 Voluntary participation

Before engaging the participants into the research study, the researcher was to ensure that participation was on their own free will. The participants also had the rights to withdraw from taking part at any time should they feel uncomfortable or threatened in the research process.

3.7.4 Confidentiality and Privacy

To ensure confidentiality, participants were to be assured that the consent form will be put separately from the questionnaire to ensure that their names are not traced back to their

questionnaires. They were also informed that the information they provide will only be accessed by the researcher and the supervisors. The questionnaires were kept in a lockable cupboard where no one will access them.

To ensure privacy, the participants were given consent forms, questionnaire, and two A4 size envelopes and they were instructed to put the documentation after completion in the separate envelopes and seal them then hand them back to the researcher. They were advised to omit their names in the questionnaires. In the case of illiterate participants, the researcher explained that they will not include the respondents' names on the questionnaire.

3.7.5 Protection of participants from any harm

Using humans as research subjects, harm and risks should be minimised whilst benefits are maximized Shannon and Raski (2009). Furthermore, any ethnic, religious, political, social, gender or other differences in a research population should be sensitively and properly handled by researchers at all stages of the research. The researcher ensured that no physical, psychological or emotional harm will occur to the participants. The researcher constructed questions in an appropriate manner that is in no way judgmental to avoid inflicting anxiety and psychological trauma during the process of responding to the questionnaire. Other possible dangers will be looked at and the research shall guard against them. Furthermore, in case some participants are harmed, the researcher will do follow up and refer the participant for counselling.

3.8 Summary

In this section, the researcher presented the research design and the system development methodology. The research design of this study was explained using a research model put forward by Oates and the paths taken involved a preliminary literature review and developing research questions. The research methodology followed was Design and Creation and the data gathering methods were observation and literature review with the data analysis method being qualitative. Different system development methodologies were studied to identify situations where each was applicable. Then based on the findings, prototyping was selected as the most appropriate development methodology for this study. The next chapter presents analysis of requirements and design of the prototype and will involve presentation of the architecture of the system, static and dynamic representations of the system among other findings.

CHAPTER 4: ANALYSIS AND INTERPRETATION

4.1 Introduction

The purpose of this chapter is to present the analysis of the results drawn from the data. The analysis and evaluation process comprise of three stages which are: gathering data, analysing data and evaluating the results of the analysis (Weaver, 2004:244).

Before developing a prototype, a statistical analysis was undertaken to determine the critical factors in the selection of the software development process, as well as to determine the most frequently used approaches. The sections below detail the statistical analysis aspects of the data.

4.2 Demographic Profile

4.2.1 Gender

The descriptive analysis focuses on the demographic profile of the respondents. From table 4.1, out of a total of 101 valid responses, 49 percent were male, while 52 percent were female. Therefore, there was a proportionate representation in terms of gender.

Table 4.1: Gender

		Frequency	**Percent**
	Male	49	48.5
	Female	52	51.5
	Total	101	100.0

4.2.2 Age and Development experience

A cross tabulation of Age and Development (Software) experience (Table 4.2) revealed the following results: a majority (52%) of the respondents were below 25 years old and had less than 1 year of experience. 12 percent who were between the ages of 26 – 35 years had between 3 – 5 years of development experience; while 3 percent of the respondents had more than 6 years of experience. Therefore, a consideration of the development experience played a critical role in determining how to proceed with the development of the prototype.

Table 4.2: Age and experience

Development Experience * Age Cross tabulation						
		Age				
		19-25	26-35	35-45	>45	Total
Development Experience	< 1 Year	52	14	3	0	69
	1 - 2 Years	0	6	1	2	9
	3 - 5 Years	2	12	4	2	20
	6 - 10 Years	0	1	0	2	3
Total		54	33	8	6	101

4.2.3 Job description

The observations in Table 4.3 were aligned to the job category of the respondents, with indications that a majority (86%) still largely involved as developers; while the remaining 14% were project managers.

Table 4.3: Job description

	Frequency	Percent
Project Manager	14	13.9
Developer	87	86.1
Total	101	100.0

4.2.4 Qualification

The qualifications of the valid responses are captured in Table 4.4; with a majority of the respondents (76%) having a Bachelor's degree; 15% have an Honours Degree; while 3% have a Master's degree.

Table 4.4: Qualification

		Frequency	Percent
Diploma		4	4.0
Undergraduate Degree		79	78.2
Honours Degree		15	14.9
Master's Degree		3	3.0
Total		101	100.0

4.2.5 Business sector / Organization sector

The question was asked to indicate the organizational sector. The most popular business sector in the study are students which represented by others on table 4.5 (58.6% - 58 participants). This was followed by system development with 38.4% with 38 participants and 3% from the manufacturing sector. Table 4.5 displays the percentage and number of respondents respectively.

Table 4.5: Frequency Table: Organization sector

Business sector		**Count**	**cumulative count**	**Valid Percent**	**Cumulative Percent**
Valid	System Development	38	37.6	38.4	38.4
	Manufacture	3	3.0	3.0	41.4
	Others	58	57.4	58.6	100.0
	Total	101	98.0	100.0	
Total		101	100.0		

4.3 Descriptive Analysis

The descriptive analysis covers the internal consistency reliability and Dimension reduction factors analysis.

4.3.1 Internal Consistency Reliability

Given the use of an instrument developed from review of literature, the focus of the descriptive analysis was to determine the reliability of the instrument and undertake normality tests. To test the reliability of the specified measurement model, the Cronbach alpha reliability coefficient was used as a measure of inter-item reliability. The measurement instrument used for measures of success had 33 items. The criteria adopted for determining reliability was according to (Landis and Koch G 1977.159-174) who indicated that if the values are between 0 - 0.2 (slightly reliable); 0.21 - 0.40 (Fairly Reliable); 0.41 - 0.60 (Moderately Reliable); 0.61 - 0.80 (Substantially Reliable) and 0.80 - 1.0(Almost Perfect). As shown in Table 4.6, the results obtained indicate that the instrument had a reliability of 0.921, which is almost perfect. This is expected, since this was a reflective instrument, which assumes that there are high correlations between the variables.

Table 4.6: Reliability Statistics

Reliability Statistics	
Cronbach's Alpha	No of Items
.921	33

4.3.2 Dimension Reduction: Factor Analysis

Given that the instrument was developed specifically for this study, an exploratory factor analysis, using Principal Component Analysis (PCA) with varimax rotation to achieve dimension reduction was used. The selection variable used was to group factors based on the responses of the 101 Project Managers involved in the study. The necessary outputs to consider in the factor analysis are the total variance explained and the rotated component matrix.

Table 4.7: Total Variance Explained

Total Variance Explained			
Component	Rotation Sums of Squared Loadings		
	Total	% of Variance	Cumulative %
1	8.549	25.907	25.907
2	8.200	24.848	50.755
3	7.316	22.171	72.926
4	6.732	20.399	93.324
5	1.826	5.534	98.858

Extraction Method: Principal Component Analysis. Only cases for which Job Category = Project Manager are used in the analysis phase.

Table 4.7 demonstrates that when the views of response are considered, then there is an explained variance of 99 percent, when 5 factors are extracted. The 5 factors extracted are depicted in Table 4.7 and the interpretations adopted follows all variables with loadings less than 0.71 were excluded. The criterion adopted was based on (Comrey A.L At al 2013), who indicated that loading values that are equal to or greater than .32 are considered to be poor; from 0.45 they are considered to be fair; 0.55 represents good loading; 0.63 as very good and 0.71 as excellent.

4.3.3 Rotated Component Matrix & Factor Re-Naming

	Component				
	1	2	3	4	5
1. User-Developer Understanding	.920				
2. Development Knowledge	.880				
3. User-Developer Trust	-.858				
4. Small Milestones	-.814				
5. Adequate Developers	.810				
6. Coverage of Development	.769				

7. Review Schedule	.767				
8. Senior Management Commitment	.729				
9. Maintenance Program	.729				
10. Training Programs		.945			
11. Appropriate Technologies		.906			
12. Straightforward Design		.875			
13. Application Domain		.747			
14. Time Management			.946		
15. User-Developer Communication			.946		
16. Intensive User Participation			.913		
17. PM Practices			.880		
18. Conflicting Requirements			.752		
19. Project Goals Agreement				.945	
20. Well Defined Project				.935	
21. Understanding Requirements				.841	
22. Complexity and Risk					-.711
23. Real Expectations					-.96

Extraction Method: Principal Component Analysis. Rotation Method: Varimax with Kaiser Normalization Rotation converged in 7 iterations. Only cases for which Job Category = Project Manager are used in the analysis phase.

From Table 4.8, the naming of factors adopted were as follows:

- Factor (Component) 1, is highly loaded with the variables: User-Developer Understanding, Development Knowledge, User-Developer Trust, Small Milestones, Adequate Developers, Coverage of Development, Review Schedule and Senior Management Commitment. The totality of these factors had an explained variance of 26 percent. The composite group of variables predominantly focuses on issues **of understanding, trust, knowledge, setting small milestones and scope of the project.** These variables are critical in enabling interactions between the developers

and the users to ensure the software development process proceeds smoothly, with minimal deviations. This group of variables can therefore be named: "**Project Excellence Enablers".**

- The second group of factors comprises of the following variables: Training Programs, Appropriate Technologies, Straightforward Design and Application Domain. These factors had a total explained variance of 25 percent, with heavy loadings characterizing all the variables. Training programs and application domain point to knowledge management success factors; while appropriate technologies and straightforward design are reminiscent of use of the right software development methodology, with simple designs. This factor may therefore be named "Knowledge Management and Design Architecture". Knowledge Management captures the two indicators of training programs and application domain; while appropriate software development methodology and straightforward design point to Design Architecture.
- The third group of factors comprised of the following variables: time management, user-developer communication, intensive user participation, PM practices and conflicting requirements. This factor had an explained variance of 22 percent and was also characterized by heavy loadings. Given the focus on various aspects of PM such as time, communication, user participation and the actual identification of other PM practices, this factor can be named: "Excellent Project Management Practices".
- The fourth group of factors comprised of three heavily loaded items related to project goal agreement, well defined project and understanding of user requirements. These components had an explained variance of 20 percent, and predominantly emphasized the ex-ante focus on developing the business value of a project. The factor may therefore be named "Business Value Proposition".
- The last factor comprised of two components which had negative loadings: Complexity and Risk; as well as Real Expectations. Further interrogation regarding the negative loadings revealed that that most of the response under estimate the complexity and risk factors associated with projects. Further, sometimes projects proceed without keeping in-sight the real expectations of the customers. Thus, by isolating these two components, the project managers are pointing to a critical need to proceed with projects by considering these two components. Given the high loading on "real expectations", this factor may therefore be named: "Realizing Real Expectations". Meeting these real expectations must be balanced by considering the "wickedness" of the project.

From Table 4.8, the dimension reduction was achieved by reducing the number of items from the original 33 to 23 items (with five latent factors). In order to carry out further analysis, composite variables were developed from the five extracted factors. The resulting five composite variables were then used as input into further regression analysis.

4.3.4 Regression Analysis

Given that the instrument was developed specifically for this study, an exploratory factor analysis, using Principal Component Analysis (PCA) with varimax rotation to achieve dimension reduction was undertaken. Five factors were extracted that formed the basis for further regression analysis. The five factors were considered as the independent variables (composite) influencing the selection and use of various software development methodologies (Waterfall, XP, RUP, JMRAD and STRADIS). These methodologies were used merely as representative approaches to form the basis for motivation for design of an unbounded rule-based architecture. The results of the regression analysis are shown in Table 4.9.

Table 4.8: Regression Analysis

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.937[a]	.879	.870	.19619

The adjusted R square value demonstrates that 87 percent of the variance in the dependent variable (software development methodology selection) can be explained by the five independent variables (Realizing Real Expectations, Business Value Proposition, Knowledge Management & Design Architecture, Excellent PM Practices and Project Excellence Enablers). Note that the equation is centred using a Centre tab stop. The high value (87%) is a good indicator of the goodness of fit of the model generated. The results of the analysis of variance (ANOVA) are presented in Table 4.9. Given that the value in the last column is less than 0.005, the inference is that the independent variables, as a set, provide a satisfactory explanation for the response variable, software development methodology selection.

The analysis of variances in Table 4.10 provide the weights of the influence of the independent variables on the response variable.

Table 4.9: Analysis of Variance

ANOVA						
Model		Sum of Squares	Df	Mean Square	F	Sig.
1	Regression	20.100	5	4.020	104.439	.000[b]
	Residual	2.771	72	.038		
	Total	22.872	77			

Table 4.10: Coefficients Analysis

Coefficients						
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	-.052	.184		-.281	.780
	Project Excellence Enablers (PM)	-.824	.124	-.550	-6.668	.000
	Knowledge Management & Design Architecture (KM)	.746	.108	.510	6.905	.000
	Excellent PM Practices (EP)	1.240	.086	.824	14.437	.000
	Business Value Proposition (BV)	-.310	.075	-.213	-4.120	.000
	Realizing Real Expectations (RE)	.161	.101	.112	1.601	.114

From the coefficient's analysis in Table 4.11, the final regression model can be inferred to be:

$$SDM\ Selection = -0.052 - 0.824PM + 0.746KM + 1.24EP - 0.310BV + 0.161RE$$

Thus, in the selection of a software development methodology (SDM), project managers must consider the inhibitors of Project Excellence Enablers (PM) and Business Value (BV) proposition. They must also consider the positive influences of Knowledge Management & Design Architecture (KM) as well as Excellent PM Practices (EP) and Real Expectations (RE).

4.4 Summary

The chapter provides the preliminary findings of a questionnaire survey based upon prior research that attempts to investigate the basic architecture of a Rule-based expert software and to identify the key factors that influence the decision of choosing an adequate software development methodology in order to develop the unbounded rule-based expert system to select a software development methodology. The findings from this chapter pointed out that the organizations do not follow the methodologies. Quantitatively, the survey presents the results analysis of the following aspects: demographic information, critical success factors, coefficients analysis, and ANOVA.

CHAPTER 5: DESIGN AND IMPLEMENTATION OF PROTOTYPE

5.1 Introduction

This chapter contains the implementation details; Interface snapshots are described, and some code snippets are also explained. Also covered is how the prototype will be tested.

The extent of success of a given project can be increased by using an appropriate Project Management Methodology (PMM) that considers the specific characteristics of the project (such as complexity, size, budget, nature of risk, etc.). To give a sense of background, Dalcher and Brodie (2007:273) maintain that a success factor is believed to be an endeavour aimed at predicting the good practices needed to accomplish the project successfully. Though the lack of success, factors do not ensure a failure, yet it can be perceived as a phenomenon that can jeopardize the project, and as a result, the project will probably fail to reach the desired results.

Practically, the project's success is normally thought of as the achievement of some pre-determined project goals, such as time, cost, and scope (Schwalbe, 2007:13). Dalcher and Brodie (2007:8) further emphasized the notion that for a project to be a success, it must meet its requirements and be completed such that it is: Within budget, Within Time, at the desired performance Level, with acceptable Quality, offering at least the minimum agreed Functionality, utilizing the assigned resources effectively and efficiently accepted by the clients.

The findings from this study provided the researcher with a rationale and a basis for the evolution of an "Unbounded Rule-Based Expert Systems Architecture" as a basis for the selection of the right software development methodology. The selection of the right PM methodology is viewed as a critical ex-ante process that can be the basis for realizing project success. The choice facing project managers in selecting an appropriate project methodology is daunting; apart from other considerations related to project characteristics such as budget, scope, schedule, performance and resource constraints common in entities in developing or emerging economies such as South Africa. Research confirms that the use of an adequate methodology for each project plays a valuable role in assuring that the project meets the objectives and goals much more efficiently and effectively. Therefore, based on the outcomes of the statistical analysis, the research proceeded to design and pilot an "Unbounded Rule-Based Expert" System which was coined "Unbounded Rule-Based Expert Systems" (URB-ES) to aid in selecting an appropriate project development methodology.

5.2 Requirements Engineering Process of URB-ES

All software systems need a base of requirements in order to be developed successfully. The requirements are a description of the system services and constraints that are generated during the requirements engineering process. The requirement engineering process is defined as the process of gathering information about the required and existing system and distilling the user and system requirements from this information (Sommerville 2011). This section contains a list of system requirements that was used to build the intelligent URB-ES prototype. These requirements were gathered through a literature review and observations. Use Cases were created out of the requirements.

The system requirements are gathered and listed on the Table 5.1, according to their specification or classification as functional and non-functional requirements.

Table 5.1: URB-ES List of Requirements

Created By	*Macheque Vhutshilo*	***Last Updated by:***	*Macheque Vhutshilo*
Date Created	*22/8/2018*	***Last Updated on***	*11/9/2018*
	Functional Requirements		**Priority**
1	The system shall be built based on the general architecture of expert-systems; thus, it shall have a knowledge base, expert model and interface.		High
2	The system shall be mobile application		High
3	The system should always respond to the user requests		High
4	The system should display the criteria to the user		High
5	The system should always accept the user requests		High
6	The system should always give immediate feedback on the user		High
7	The system shall always choose the best suitable methodology according to the user specification		High

Table 5.1: continued

8	The system shall be fed with user requirements	High
9	The system shall suggest the best	Low
	Non-Functional Requirements	
10	The system shall be mobile-based.	High
11	The system shall have task-focused interfaces.	High
13	The system shall have as few interfaces as possible to enhance ease of use.	High
14	The response time shall be as minimal as possible.	medium
15	The system shall not consume many resources such as memory and processor time.	medium

5.3 High Level Architecture of URB-ES

For URB-ES prototype, the Model View Architecture in Figure 5.1 was followed. This architecture contains three components, namely **The Model, The View and The Controller**. The central component of MVC is **The Model**. This component captures the behaviour of the application in terms of its problem domain, independent of the user-interface (Burbeck 1992). **The Model** directly manages the data, logic and rules of the application, java classes make up the model of the proposed system.

The View can be any output representation of information, such as a chart or a diagram. With the URB-ES, **The View** was created using xml.

The Controller is responsible for accepting inputs and converting them into commands for The Model or The View (Rj45 2008). In the URB-ES, this component includes activity classes and other java classes that contain navigation code linking the view and the model and general control of execution flow.

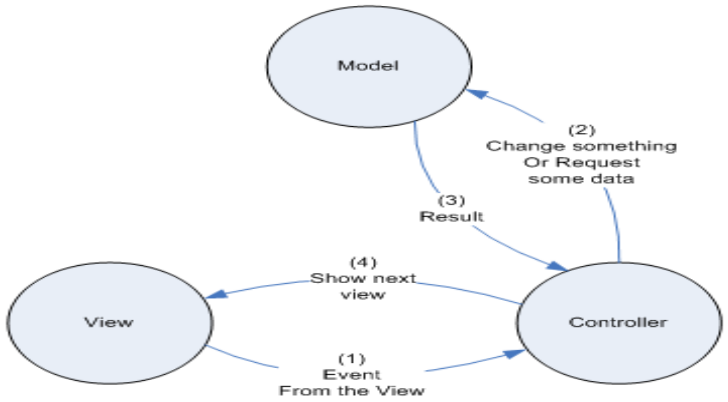


***Figure 5.1**: MVC architecture (Brown et al. 2008)*

5.3.1 Use case descriptions

Figure 5.2 represents Use Cases for ease of understanding by the developer. A Use Case diagram is a graphic depiction of the interactions among the elements of a system.

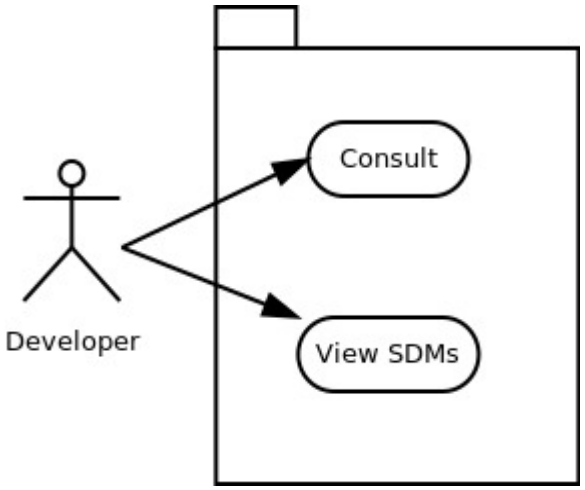


***Figure 5.2**: Use Case Diagram*

Table 5.2: Use Case Description of Consultation between User and URB-ES

USE CASE ID	URB-ES		
Use Case Name	**User consult**		
Created By	Macheque Vhutshilo	**Last Updated By**	Macheque Vhutshilo
Date Created	08/11/2018	**Last Revision date**	10/12/2018
Actors	User, URB-ES		
Description	The user consults and the SDM give the responds according to user specification.		
Trigger	User consult		
Preconditions	1. The best methodology will be displayed. 2. The SDM respond to user		
Post Conditions	1. The best methodology will be selected according to user specification		
Normal flow	1. User consult 2. The SDM model consults the expert model. 3. The expert model weights the criteria. 4. The SDM selects the best methodology.		
Alternative flow	The expert model ways to find best methodology 1. Proper feedback is given.		

The Use Case describes the main process and the interaction between the user and URB-ES system. Table 5.2 describes the processes that will take place during consultation between the user and SDM system. The user will run the URB-ES system and select to consult, URB-ES will display selection criteria. The user will therefore select one or more criteria. URB-ES will display a form where the system weights each, and the user will then enter weights, the system brings SDM, first being the best.

5.3.2 Architectural solution.

The architecture in Figure 5.3 shows how best the URB-ES can solve the problem that the software developers are currently facing as they have a large pool of methodologies to choose from. The process designed in Figure 5.3 was used as a basis for the design of an unbounded rule-based software development methodology. Given that Project Managers are faced with a large pool of generic project management methodologies and numerous constraints to consider, the flow chart can document the decisions that they face as they go through the

process of making a choice. Documenting the process clarifies the steps that may need to be automated.

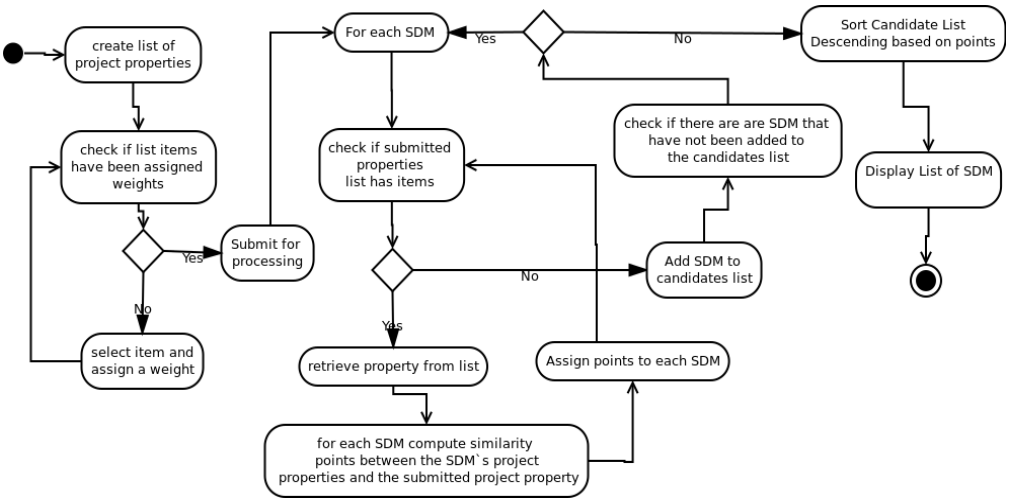


Figure 5.3: *Architectural Diagram*

5.3.3 The Sequence Diagram

In this section, the researcher presents dynamic representations of URB-ES object communications under different scenarios. A sequence diagram in Figure 5.4 shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between Objects needed to carry out the functionality of the scenario.

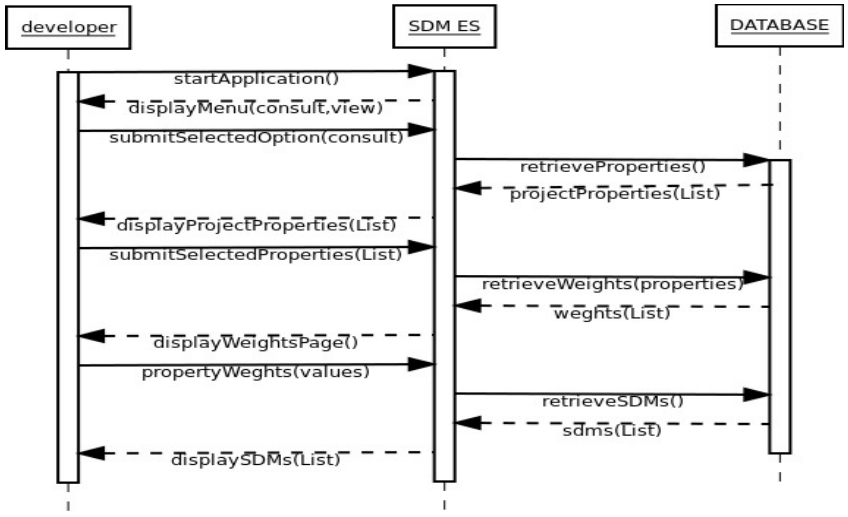


Figure 5.4*: Sequence Diagram*

5.4 Technical Implementation of the URB-ES

URB-ES is a mobile application that runs on android platform. The application was developed using the Android ADT bundle. The package comes with an eclipse IDE and other

development tools like an emulator. Java is the language that was used together with xml for interfaces. The database that was used was SQLite, which is a database that can be integrated into any android application and uses SQL commands.

5.4.1 Application

The URB-ES application is packaged as an apk file, the file format for android applications. Figure 5.5 shows 'URB-ES application icon when installed on an android device in which the user can interact with. For demonstration purposes, the application runs on a nexus 7, android 4.2 (Jelly Bean).

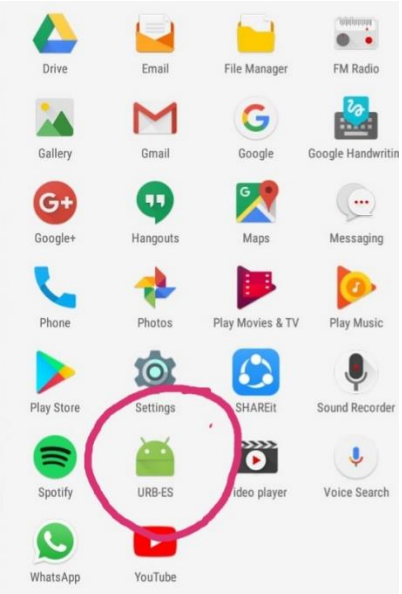


***Figure 5.5**: URB-ES shortcut on home screen*

5.4.2 Consultation process with the URB-ES

Figure 5.6 displays the main menu of URB-ES application which gives the user three options (Consult, Views SDMs and Help) to choose from. In case the user would like to consult, by clicking the "Consult" button in Figure 5.6, a pop-up window displaying the URB-ES list of criteria to be selected from is shown Figure 5.7.

***Figure 5.6**: URB-ES main Manu*

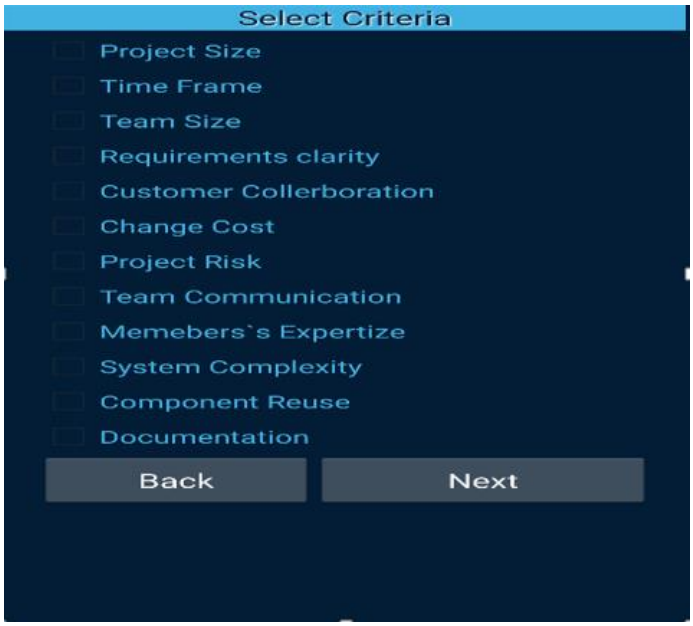


***Figure 5.7**: List of criteria*

Figure 5.8 shows the seven user's selected criteria for the project requirements in hand.

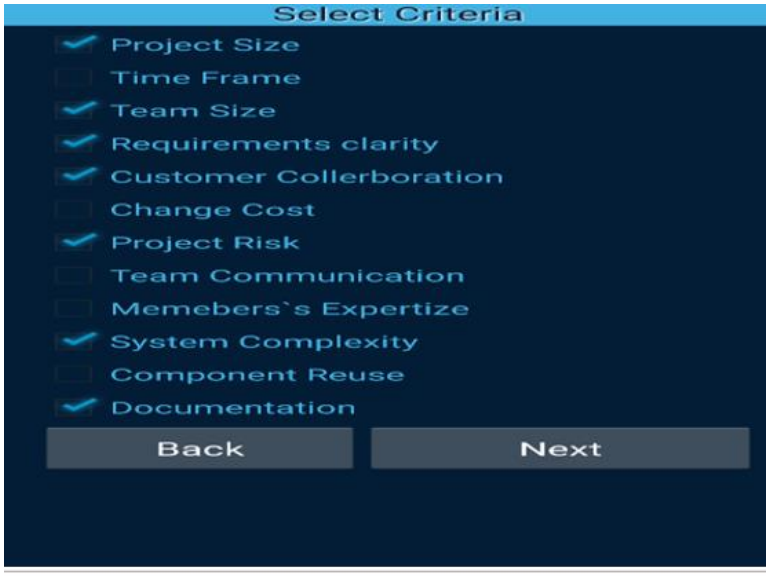


Figure 5.8: *Project requirements*

5.4.2.1 Mapping Selection Factors to Criteria

In developing the criteria for starting the consulting process in Figure 5.7 and Figure 5.8, the factors that were identified during the statistical analysis form the basis for the criteria as shown in table 5.3. The considered factors were Project Excellence Enablers, Knowledge Management, Business Value Proposition, Excellent Project Management Practices and Realizing Real Expectations. Table 5.3 maps these identified factors from the quantitative analysis to the developed criteria. The mapping of factors to the criteria confirms that the application adhered to the needs of the users as expressed by the survey respondents.

Table 5.3: *Mapping Selection Factors to Criteria*

Selection Factor	**Criteria**
Project Excellence Enablers	Requirements Clarity, Changing Cost
Better Knowledge Management	Components Reuse, Documentation
Excellent PM Practices	Time Frame, Team Communication, Members Expertise, Team Size
Business Value Proposition	Customer Collaboration
Realizing ‘Real’ Expectations	System Complexity, Project Risk, Project Size

5.4.2.2 Weighting in URB-ES

Based on Figure 5.8 selected criteria the reasoning approve Zaied et al (2013) weighting principles. The URB-ES uses the if then else statement to assign point values to each SDM e.g. if: user requirements clarity is low .Then: check SDMs property and assign them 0; 0.5 or 1.0 points for Small (0.0), Medium (0.5) or High (1.) respectively as shown in Figure 5.9.

***Figure 5.9**: Weights for SDM*

5.4.2.3 List of Most Appropriate Methodologies

Figure 5.10 shows the methodologies displayed according to project specification. The list is displayed from the most to the less appropriate depending on the project at hand. This will help the developer to select the most appropriate methodology for that project. In this case, Waterfall model is the best for this project followed by Extreme programming, Spiral on that order.

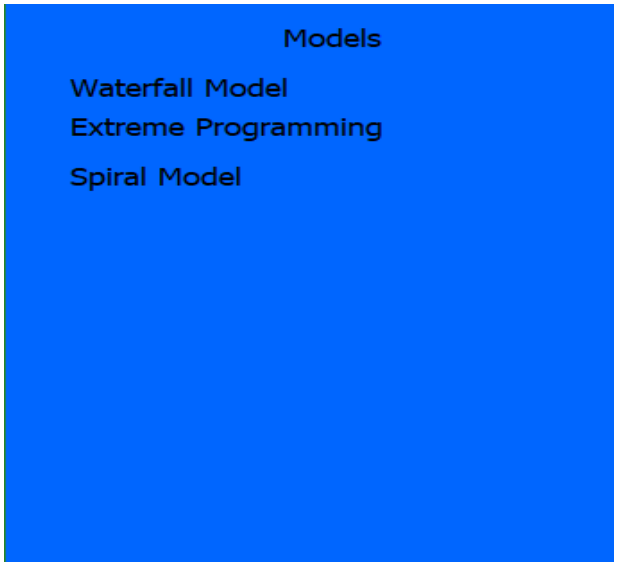


Figure 5.10: *Displayed SDMs*

5.4.3 Viewing the SDMs in URB-ES

Assume that the user would like to view SDMs within the URB-ES Systems, by clicking the view button a pop-up windows Figure 5.11 displays the list of software development methodologies available. Some of these software development methodologies were discussed in Chapter 2 subsection 2.4 in details.

Figure 5.11*: Window Displaying SDM*

5.5 Testing

Throughout the system development, Junit was used for automated method testing and inspection was also used to discover more bugs and to improve on code efficiency. For testing the URB-ES, 15 computer science students were asked to install and use the prototype on their android devices. Honours students were chosen because this is the level where students fully understand Object-oriented Programming and have solid understanding of software development. Feedback from these students was used to constantly refine the URB-ES until its current state. Feedback was in the form of comments in usability, bugs in the system, resource consumptions and other performance issues.

However, for the test results to be completely reliable the system still needs to be tested in the industry using real requirements for the project on the hand. It would be helpful to see how best the URB-ES solves the problem that the project managers are currently facing.

The researcher would have had two randomly select groups; one using both project managers and students to generate more results and how accurate the URB-ES could solve their challenges, this could have given the researcher more reliable results but due to time constraints it was not possible.

5.6 URB-ES evaluation

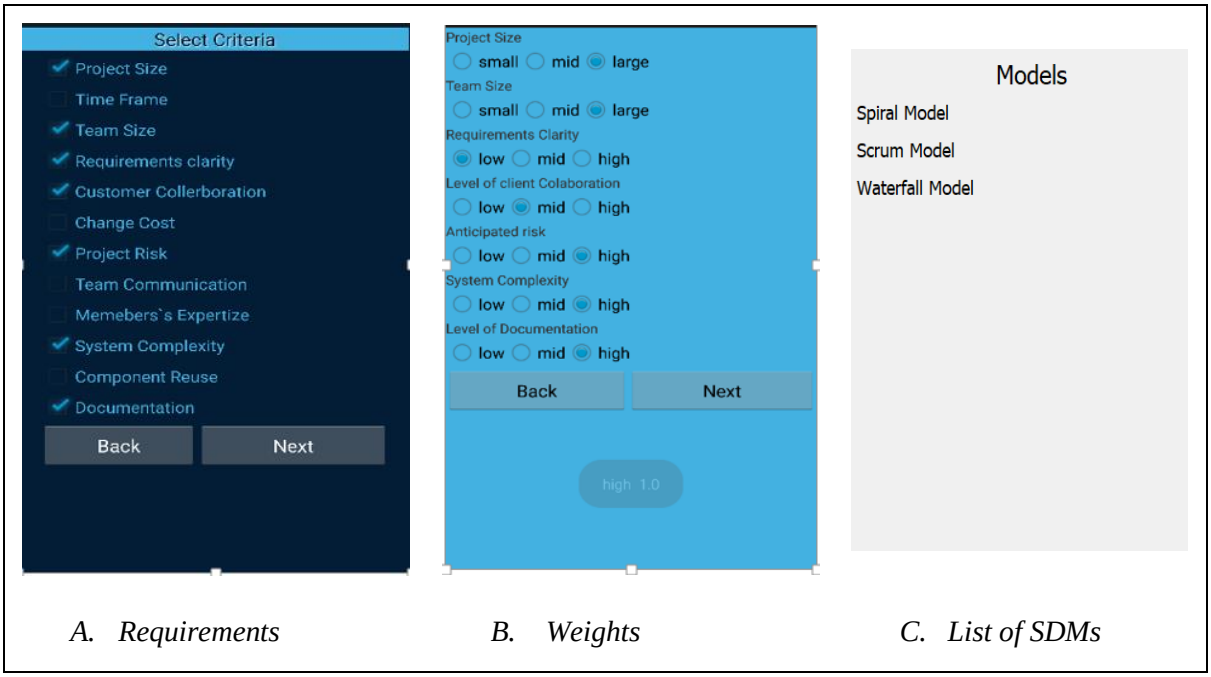
Assume that the user has the following requirement for the project:

i. **Scenario 1:**

Complexity = High; Project Size = High; Team Size = High' Requirements Clarity = Low; Client collaboration = Medium; Risk = High; Documentation = High

Which methodology does the URB-ES application recommend?

The list of SDMs is displayed from the most appropriate to the least depending on the project at hand as shown in **Figure 5.12**.



A. Requirements *B. Weights* *C. List of SDMs*

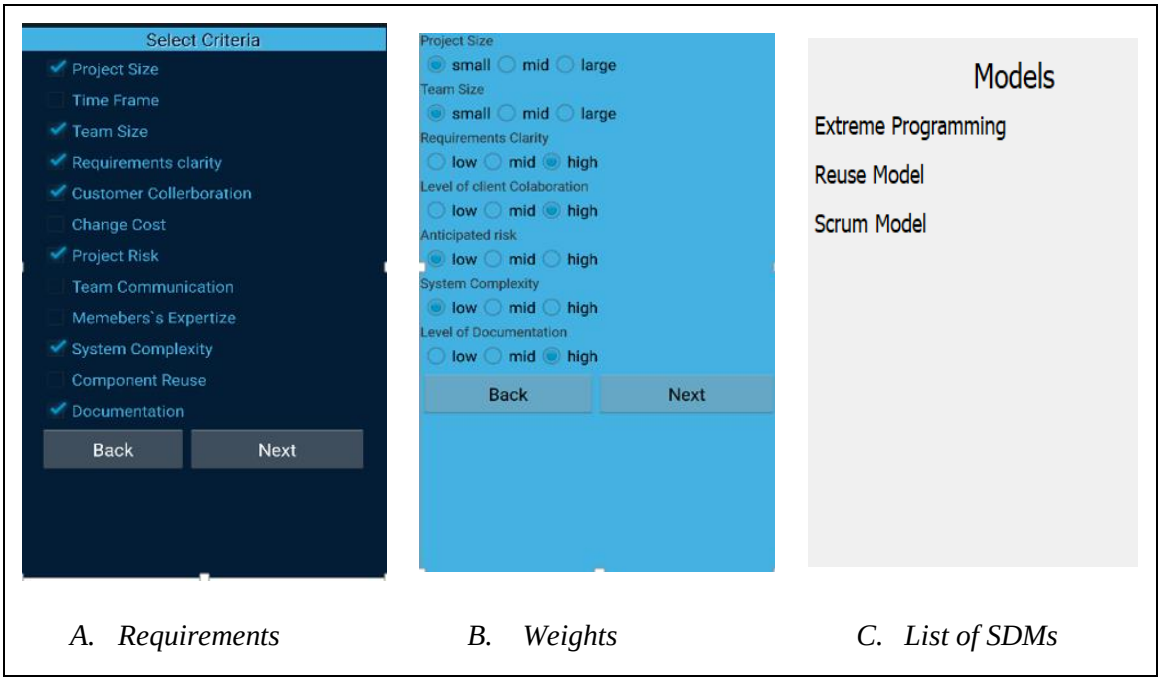
Figure 5.12: *URB-ES Evaluation Scenario 1*

The URB-ES will recommend Spiral Model followed by Water Fall and then Scrum model based on the nature of the project and given requirements. The spiral model is a risk-driven software development process model. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping. Looking at some factors such as requirements clarity = low, Risk = High and complexity is high; this project will have lot of requirements changes. Chapter 2, subsection 2.4.4 of the spiral model is explained in detail.

ii. Second Scenario:

Assume that we have a small project whereby Complexity = Low; Project Size = Low; Team Size = Low; Requirements Clarity = High; Client collaboration = High; Documentation = Low

Which methodology does the application recommend?



A. Requirements *B. Weights* *C. List of SDMs*

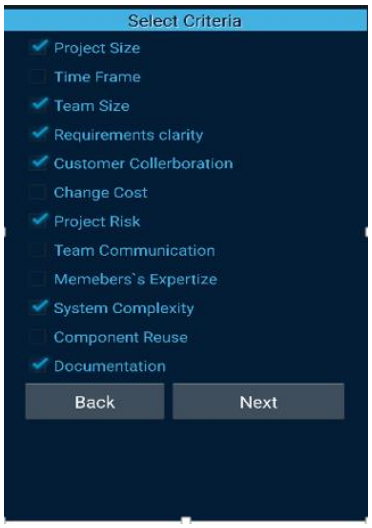
***Figure 5.13**: URB-ES Evaluation Scenario 2*

The URB-ES in Scenario 2 Figure 5.13 recommend Extreme Programming followed by Reuse and then Scrum Model based on the nature of the program.

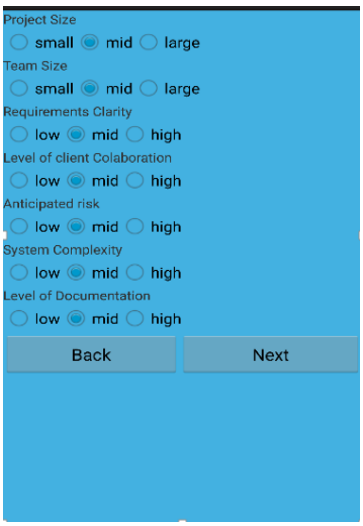
iii. Scenario 3:

All variables are medium.

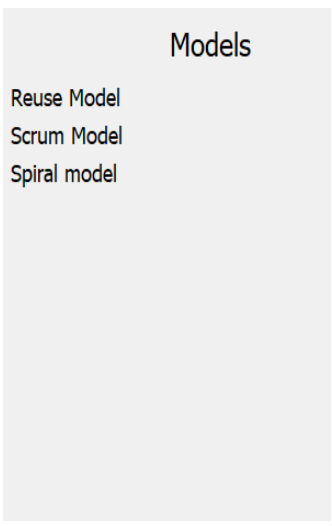
Complexity = Medium; Project Size = Medium; Team Size = Medium' Requirements Clarity = Medium; Client collaboration = Medium; Risk = Medium; Documentation = Medium



A. Requirements



B. Weights



C. List of SDMs

Table 5.4: URB-ES Evaluation Scenario 3

The URB-ES in Scenario 3 Figure 5.14 recommend Reuse followed by scrum and then Scrum Model based on the nature of the program.

5.7 Summary

This chapter presented the technical implementation of the URB-ES system. Interfaces and code snippets were presented and explained, also described is the Use Case and Activity diagram of the system. A brief outline of how testing was carried out was also given. The next chapter presents the conclusion and future work.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter focused on the conclusions from the entire study. The basis of discussions is the answers to the research questions formulated at the beginning of the study. The chapter would subsequently identify and discuss the limitations and suggest future work to give further innovative solutions with respect to the best practices of the deployment of software development projects. Finally, the recommendations were presented based on the nature and the outcomes of the study.

6.2 Research Aim and Main Findings of the Study

This research presented the modelling and development of an unbounded rule based expert system for selecting a suitable software development methodology according to software project features. By combining rule-based knowledge representation with object-oriented database modelling, a flexible and extensible prototype expert System was developed.

One of the objectives was to investigate the main features of SDM software; this was done mainly through a literature review. The main components of expert-systems plus some of the existing URB-ES software were investigated. After identifying the components of each of these SDMs, we identified the common features and concluded came up with the main features on SDM software when developing a prototype.

URB-ES prototype is a mobile-based application that was developed in Java to run on Android devices. We wished that the application had a local database and then created a central web database that would constantly update the local one. However, due to limited time, the researcher only managed to create a local database using SQLite.

Even though the researcher had some development challenges, the researcher successfully implemented a solution that supports the go-green environment and the prototype was successfully developed.

6.3 Summary of findings

A summary of the findings relating to each objective of the study is provided below.

6.3.1 Investigate the Architecture of a Rule-Based Expert System

One of the objectives was to investigate the main features of SDM system. This was done through a literature review. We first investigated the main components of expert-systems then

went on to investigate some of the existing SDM software. After identifying the components of each of these SDMs, we identified the common features of these and concluded that these were the main features on SDM software which must be kept in mind when developing our prototype.

6.3.2 Identify Factors that Influence the Choice of a Software Development Methodology

A survey was conducted focusing on identifying the factors that influence the choice of SDM. A quantitative analysis using factor analysis and regression analysis was conducted to identify these factors. From the analysis in Chapter 4 subsection 4.3.3, two critical inferences can be made. The first relates to the group of factors that are linked to Project Excellence Enablers (PE). From the factor analysis in Table 4.7, PE, as a composite variable comprised of variables related to User-Developer Understanding, Development Knowledge, User-Developer Trust, Small Milestones, Adequate Developers, Coverage of Development, Review Schedule, Senior Management Commitment and Maintenance Program. The interpretation from the results of the regression model is that, if these factors are absent or not considered in the choice of a software development methodology, then their influence is negative, thus affecting the software development process negatively. This group of factors were christened Project Excellence Enablers, with the connotation that, when embarking on a software development process, key factors related to project quality should be considered ex ante. This inference finds support in recent literature, with research maintaining that despite advancements in PM processes, tools and systems, project success has not significantly improved (Bredillet et al. 2018). This problem raises issues related to the effectiveness of the PM process, thus the focus of this project on designing an unbounded rule-based software development architecture to enhance PM practice.

The second inference from the regression results focuses attention on the second negative group of factors christened Business Value Proposition (BV). This group of factors was early theorized and confirmed to comprise of the following variables from the factor analysis results: Project Goals Agreement; Well Defined Project and Understanding Requirements. Again, the focus of these variables, from their meaning, is on the ex-ante considerations in the project management process. The implications are that for a successful software development process, key considerations must take into account project goals agreement amongst the stakeholders; proper elucidation of the project and an understanding of the requirements of the project. The issue of business value proposition in projects remains problematic and the outcomes from this study illustrate this continuing dilemma. For instance, (Smyth et al .2018) contends that the framing of the value proposition requires knowledge of the stated content,

the underlying business case and problem the project is addressing. However, project failure is still rife despite improvements in project management processes, tools and techniques

The findings from this study therefore provided a rationale and a basis for the evolution of an "Unbounded Rule-Based Expert Systems Architecture" for the selection of the right software development methodology. The selection of the right PM methodology is viewed as a critical ex-ante process that can be the basis for realizing project success. The choice facing project managers in selecting an appropriate project methodology is daunting; apart from other considerations related to project characteristics such as budget, scope, schedule, performance; and resource constraints are common in entities in developing or emerging economies such as South Africa. The research confirmed that the use of an adequate methodology for each project plays a valuable role in assuring that the project meets the objectives and goals much more efficiently and effectively. Therefore, this project seeks to contribute to making decision making in project management better through the development of an "Unbounded Rule-Based Expert Systems" that can aid in selecting an appropriate project development methodology. This would be a critical innovation for project decision makers by increasing project success rate, minimizing costs and risks and assuring performance.

6.3.3 Develop a Software Prototype of a Rule-Based Expert System

With the identified factors and components in mind, we successfully designed and implemented the task focused SDM system. The prototype is a mobile-based application that is developed in Java to run on android devices.

6.4 Recommendations

We recommend that studies about the conceptualization and operationalisation of SDMs need to be conducted to equip practitioners with knowledge in their endeavour to carry out IT projects; since the study discloses that, for some stipulated reasons, there are still practitioners who do not employ SDM practices. The other fact is that the practices of SDMs are always associated with the development of systems (Duggan & Reichgelt, 2006:9), hence it is vital to dig deeper regarding the concept of SDMs.

6.5 Limitations and future work

- Some limitations exist in that the impact of method-in-action to project success was not fully studied. It can be a good topic for future research. This concept is underpinned by Fitzgerald (2002).

- Future research should scrutinize the concept that employment of SDMs practices eliminates or takes away the authorities and responsibilities from the practitioners. According to Fitzgerald et al. (2002:124) JSD was aimed at minimizing the responsibility of the developer in the development process.
- Future studies should seek to find out the risks associated with the adoption of SDMs. Duggan and Richest (2006: xv) point out that the deployment of SDMs is associated with some risks.
- If possible, similar research should be conducted periodically because it provides knowledge to the practitioners who are interested in the domain of software development and will make them aware of potential problems and proposed solutions.
- The nature and size of projects utilizing systems development methodologies practices should be investigated more precisely to determine how they affect IT project success.
- The study results have failed to find evidence as to whether systems development methodologies have a direct impact on the success of IT projects. As a result, future studies could measure the impact of particular SDMs. Specifically, according to Chow and Cao (2008:961) research into agile methods is still limited in academics circles.
- The results of this study suggest a need to discover the realities involved in the usage and substantiation of methodologies with regard to IT project success in future. Dyba et al. (2005:447) conducted a study to measure methodology usage and this concept needs to be further examined.
- Future research should scrutinize the concept that employment of SDMs practices eliminates or carries away the authorities and responsibilities from the practitioners. According to Fitzgerald et al. (2002:124) JSD was aimed at minimizing the responsibility of the developer in the development process.
- Future studies should seek to find out the risks associated with the adoption of SDMs. Duggan and Richest (2006: xv) point out that the deployment of SDMs is associated with some risks.

The product of this work is an incomplete prototype that the researcher intends to continue perfecting. Work to be done includes:

- The URB-ES must have online database instead
- The URB-ES must be platform independent so that it run at any operating devices

6.6 Summary

The findings from this study therefore provided a rationale and a basis for the evolution of an "Unbounded Rule-Based Expert Systems Architecture" as a basis for the selection of the right software development methodology. The selection of the right PM methodology is viewed as a critical ex-ante process that can be the basis for realizing project success. The choice facing project managers in selecting an appropriate project methodology is daunting; apart from other considerations related to project characteristics such as budget, scope, schedule, performance; and resource constraints common in entities in developing or emerging economies such as South Africa. Research confirms that the use of an adequate methodology for each project plays a valuable role in assuring that the project meets the objectives and goals much more efficiently and effectively

We provided answers to the research questions which were considered to be the main objectives of this study. The demographic information and the reasons for not using SDMs results were in supplementary fashion summarized and interpreted. It was found that, to some extent, there is a relationship between systems development methodologies (as we saw, high numbers of respondents agree that those quality and process factors influenced by SDMs improve the development of IT projects). Most of these factors have a positive correlation with IT project success. Nonetheless, an SDM factor is not dependent on or responsible for the success of IT projects. A high number of practitioners and projects utilize the practice of SDMs. On the other hand, it is recognized that for reasons such as competency of current method, SDMs were not used, and to some degree, SDMs were adapted to fit business needs.

The top critical success factors which contribute to IT project success were also investigated and outlined. It was indicated that the most popular factor was the identification of potential benefit and risks.

On average, the study confirms that there is a relationship between systems development methodologies (SDMs) and IT project success

REFERENCE LIST

A. L. Comrey and H. B. Lee, A first course in factor analysis (2nd ed.). 2013.

Al Ahmar, M.A., 2005. Rule based expert system for selecting software development methodology. *Journal of Theoretical and Applied Information Technology*, pp.143-148.

Anon, 2009. Introducing Android.

Anon, 2011. Adobe Flex 4.6 And Adobe Flash Builder 4.6.

Anon, 2013a. Mobile Web Apps vs . Mobile Native Apps : How to Make the Right Choice. Available at: www.lionbridge.com.

Anon, 2013b. Object oriented systems development methodology. In pp. 2–4.

Anon, 2014a. 10 Hot Consumer Trends 2014.

Anon, 2014b. Adobe Air. Available at: http://www.adobe.com/in/products/air.html [Accessed july 10, 2017].

Anon, 2014c. Computer Basics: Understanding Applications. Available at: http://www.gcflearnfree.org/computerbasics/3 [Accessed October 1, 2017].

Anon, 2014d. Most common SDK versions. Available at: http://www.appbrain.com/stats/top-android-sdk-versions.

Anon, 2014e. The Architecture of ActiveMath. Available at: http://www.activemath.org/Software/Achitecture [Accessed October 10, 2017].

Anon, 2014f. Titanium SDK. Available at: http://www.appcelerator.com/platform/titanium-sdk/ [Accessed October 9, 2017].

Bern, A., Nikula, U., Pasi, S. & Smolander, K. 2007. Contextual factors affecting the software development process: an initial view. (2nd AIS SIGSAND European Symposium on Systems Analysis and Design. Gdansk, Poland, 5 June 2007.

Braun, V. and Clarke, V. (2006). *Using Thematic Analysis in Psychology. Qualitative Research in Psychology.* [Online] 3 (2). p. 77-101. Available from: <http://dx.doi.org> [Accessed 3 December 2014].

Bredillet, C., Tywoniak, S., & Tootoonchy, M. (2018). Exploring the dynamics of project management office and portfolio management co-evolution: A routine lens. *International Journal of Project Management*, *36*(1), 27-42.

Buchanan, B.G. and Duda, R.O., 1983. Principles of rule-based expert systems. *Advances in computers*, *22*, pp.163-216.

Burbeck, S., 1992. How to use Model-View-Controller (MVC). Available at: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html [Accessed October 22, 2017].

Christenson, D. 2007. The Role of Vision as a Critical Success Element in Project Management A Thesis Submitted in partial completion of the requirements for a Doctor of Project Management Royal Melbourne Institute of Technology University Submitted By, 384.

Cockburn A. Agile software development: agile software development series. Addison-Wesley, Boston, 2002

Computer Basics: Understanding Applications. (2014).

Dalal, A.F., 2011. The 12 pillars of project excellence: a lean approach to improving project results. CRC press.

Dalcher, D. & Brodie, L. 2007. Success IT projects. London: Thomas Learning.

Dodig-crnkovic, G., Scientific Methods in Computer Science.

Durga, P. T. 2014. Top 5 android Development Tools.

Durga, P.T., 2014. Top 5 android Development Tools.

Dybå, T., & Dingsøyr, T. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

European Commission Directorate General for Regional Policy and Cohesion. (1998). Understanding and Monitoring the Cost-Determining Factors of Infrastructure Projects: A User's Guide. *European Commission*.

Folorunso, I. O., Abikoye, O. C., Jimoh, R. G., & Raji, K. S. 2012. A Rule-Based Expert System for Mineral Identification. *Journal of Emerging Trends in Computing and Information Sciences*, *3*(2), 205–210.

Gaševic, D., Djuric, D. and Devedžic, V., 2006. *Model driven architecture and ontology development*. Springer Science & Business Media.

Geambaşu, C. V., Jianu, I., Jianu, I., & Gavrilă, A. 2011. Influence factors for the choice of a software development methodology. *Accounting and Management Information Systems*, *10*(4), 479–494

Griffin, A.S. & Brandyberry, A.A. 2008. System development methodology usage in industry: a review and analysis. Journal of information system applied research (JISAR), 3(19):1-18

Hartley, R. T. 1985. Expert Systems Methodology : A Conceptual Analysis, *1*, 11–22.

Highsmith A. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY: Dorset House Publishing, 2000

Highsmith J. Agile Software Development Ecosystems. Addison Wesley, 2002.

Huisman, H. & livari, J. 2006. Deployment of systems development methodologies: perceptual congruence between IS managers and systems developers. Information & management, 43:29-49.

Investigation, I. 2008. Selecting a development approach, 1–10.

J. R. Landis and G. G. Koch, “The measurement of observer agreement for categorical data,” *Biometrics*, pp. 159–174, 1977.

Jahns, R.-G. 2010. 6 reasons why mobile apps will become as important for companies as corporate websites.

Jahns, R.-G., 2010. 6 reasons why mobile apps will become as important for companies as corporate websites.

Joussen, A.M., Poulaki, V., Le, M.L., Koizumi, K., Esser, C., Janicki, H., Schraermeyer, U., Kociok, N., Fauser, S., Kirchhof, B. and Kern, T.S., 2004. A central role for inflammation in the pathogenesis of diabetic retinopathy. The FASEB journal, 18(12), pp.1450-1452.

Kok, J.N., Boers, E.J., Kosters, W.A., Van der Putten, P. and Poel, M., 2009. Artificial intelligence: definition, trends, techniques, and cases. *Artificial intelligence*, *1*.

Lee, I. 2013. *Mobile Applications and Knowlegde Advancements in E-business*.

Loggia, C., & Basili, V. (1986). Expert Systems for Software Engineering Management: A Summarized Evaluation.

Madsen, S., Kaut, K. & Vidgen, R. 2006. A framework for understanding how a unique and local IS development method emerges in practice. European journal of information systems, 15:225-238

Masrek, M., Hussin, N. & Tarmuchi, N. 2008. An exploratory study on systems development methodologies for web-based applications. Information management & computer security, 16(2):137-149.

McConnell S. Rapid development: taming wild software schedules. Microsoft Press, Redmond, Wash, 1996.

Meso, P., Madey, G., Troutt, M.D. & Liegle, J. 2006. The knowledge management efficacy of matching information systems development methodologies with application characteristics: an experimental study. Journal of systems and software, 79:15-28

Mihailescu, D. & Mihailescu, M. 2009. Exploring the nature of information systems development methodology: a synthesized view based on a literature review. (Proceedings of the Fifteenth Americas Conference on Information Systems, San Francisco, California August 6th-9th 2009 by an authorized administrator of AIS Electronic Library (AISeL))

Oates, B.J., 2005. *Researching Information Systems and Computing*.

Omland, H.O. & Nielsen, P.A. 2009. Actors‟ competencies or methods? A case study of successful information systems development. (20th Australasian Conference on Information Systems. 2-4 Dec 2009, Melbourne).

OSQA, 2009. SDLC Models.

Qian, X., Zhu, G. & Li, X., 2012. Comparison and Analysis of the Three Programming Models in Google Android.

Qian, X., Zhu, G., & Li, X. 2012. Comparison and Analysis of the Three Programming Models

in Google Android.

Report, S. et al., 1992. A Reuse-Based Software Development Methodology. , (January).

Rj45, 2008. Simple Example of MVC (Model View Controller) Design Pattern for Abstraction - CodeProject.

Rollins, M. & Sandberg, R., 2013. *The business of Android Apps Development.*

Rollins, M., & Sandberg, R. 2013. *The business of Android Apps Development*.

Rosemarie, A. 2007. *Thematic Content Analysis (TCA). Descriptive Presentation of Qualitative Data.* [Online]. Available from: <www.wellknowingconsulting.org> [Accessed 10 June 2017]

Schwalbe, K. 2007. Information technology project management. 5th ed. Boston, Mass.: Thomson/Course Technology.

Seflek A., and Carman K., (2010), "A design of an expert system for selecting pumps used in agricultural irrigation", Mathematical a

Shrivastava, P., Satpathy, S.K. and Nagwanshi, K.K., 2011. Implementation of an Expert System as Spiritual Guru for Personality Development. *International Journal of Computer Theory and Engineering*, *3*(1), p.116.

Smyth, H., Lecoeuvre, L., & Vaesken, P. (2018). Co-creation of value and the project context: Towards application on the case of Hinkley Point C Nuclear Power Station. *International Journal of Project Management*, *36*(1), 170-183.

Sommerville, I., 2011. *Software Engineering Ninth Edition*,

Standish Group International. 1995. Standish group report. Chaos, T23E-T10E Boston: The Standish Group International. http://www.scs.carleton.ca/~beau/PM/ Standish-Report.html. Date of access: [November 2018]

Vavpotič D., Vasilecas O. Selecting a Methodology for Business Information Systems Development: Decision Model and Tool Support. ComSIS, 2012,

Wild, J. and Diggines, C. 201. *Marketing Research.* 2nd Edition. Juta and Company: Cape Town.

Yahya, Y., Yusof, M.M., Yusof, M. & Omar, N. 2002. The use of information system development methodology in Malaysia. Jurnal Antarabangsa (Teknologi Maklumat), 2:15-34.

Young, D. C. 2013. Software Development Methodologies. Alabama Supercomputer Center (ASC.[Online].Availableat:https://www.researchgate.net/publication/255710396_Software_Development_Methodologies. [Accessed 28/10/2017].

Zaied, A. N. H., Ibrahim, S., & Aal, A. 2013. Rule-based Expert Systems for Selecting Information Systems Development Methodologies, (August), 19–26. https://doi.org/10.5815/ijisa.2013.09.03

APPENDIX 1: Questionnaires Survey Form

SECTION A: Research questionnaire:

The purpose of this study research is to investigate the basic architecture of Rule-based expert software and to identify the key factors that influence the decision of choosing an adequate software development methodology in order to develop unbounded rule-based expert system to select a software development methodology

Your participation in this study is most appreciated and the rest assured that your answer will be kept confidential and the completion of this questionnaire is voluntary.

The results of this study will be provided on request.

1. Please indicate your gender	
1.1 MALE 1.2 FEMALE	1 2

2 Please indicate your age	
2.1. 18 or less	1
2.2. 19 -25	2
2.3. 26 -35	3
2.4. 36 -45	4
2.5. 45 or above	5

3 Please indicate your job category	

3.1 Project Manager/Leader 3.2 Developer 3.3 Other, please specify	1 2 3 ……

4 Indicate your qualification	
4.1. senior certificate (High school) 4.2. certificate or Diploma 4.3. university or Technicon 4.4. Honours degree 4.5. Master's Degree 4.6 doctoral degree 4.7. other, please specify ……	1 2 3 4 5 6 7

5. Please indicate the sector of your organization	
5.1. System development 5.2. Manufacture 5.3. financial, banking 5.4. government 5.5. Other, please specify ……..	1 2 3 4 5

6. please indicate the size of your organization	
6.1. 1-50 employees 6.2. 51-200 employees 6.3. 201-500 employees	1

6.4. 501-1000 employees 6.5. more than 1000 employees	2 3 4 5

7. Please indicate your personal experience in system/ software development	
7.1. None	1
7.2. less than 1 year	2
7.3. 1 -2 years	3
7.4. 3 – 5 years	4
7.5. 6 -10 years	5
7.6 11 years and more	6

8. Please indicate the size of last information systems project you were involved	
8.1. Small	1
8.2. Medium	2
8.3. large	3
8.4 very large	4
8.5 I have not participated in any software/ information system development project	5

9 Please indicate how long it took to complete the last project you were involved with	

9.1. less than a year	1
9.2. 1 -2 years	2
9.3. 3 -5 years	3
9.4. 6 years or more	4

SECTION B: Success Factors Influencing information systems development

On the lines below, place an X to indicate the factors used successfully in developing the system in your department

To what extent do you agree with the following statements as a valid description of the last information systems development project you were involved with?	Not at all	To a little	To some extent	To a greater extent
1The project used an appropriate systems development methodology	1	2	3	4
2The project had clear defined system goal and objectives	1	2	3	4
3 The project had a simple straightforward design	1	2	3	4
4 The project had good training programs for all involved.	1	2	3	4
5. The project had a review schedule after project completion.	1	2	3	4
6. The project had well defined and organized maintenance program.	1	2	3	4
7. The project use an appropriate hardware and software technologies	1	2	3	4
8. The project recognized the level of complexity and risk.	1	2	3	4
9. project was well defined	1	2	3	4
10. During the project, users were aware of the requirements	1	2	3	4
11. During the project user's requirements were changed all times.	1	2	3	4
12 developers understood user's requirements well.	1	2	3	4

13. The was general agreement on project goal and objectives	1	2	3	4
14.different user's department or users had conflicting requirements	1	2	3	4
15 the project had user's expectations that are real.	1	2	3	4
16 During the project, developers were familiar with technology-based structure	1	2	3	4
17. The project had the developed which was very complex	1	2	3	4
18. The project had intensive user's participation.	1	2	3	4
19. Senior management was highly committed to the project	1	2	3	4
20. Project had a good project management practices.	1	2	3	4
21. Smaller project milestones	1	2	3	4
22. The project had good user developer trust	1	2	3	4
23. the project had good user-developer communication	1	2	3	4
24. The project had good user developer mutual understanding.	1	2	3	4
25. The project had adequate time for development	1	2	3	4
26. The developer had adequate developer available to work on the project.	1	2	3	4
27. The project had a good developer's technical expertise.	1	2	3	4
28. The project had good developer of the application domain.	1	2	3	4
29. Developers Knowledge of the system development process was good	1	2	3	4
30 The project covered the entire development life cycle.	1	2	3	4
31. The project had regular recording of any activities, mistake, improvement occurred during development phase.	1	2	3	4
32. The project had clear and specified business priorities.	1	2	3	4
33. The project had a clear vision and objectives	1	2	3	4

34.Ifother.Please specify……				

SECTION C: **The role of system development methodology towards software development project success**

On the lines below, place an X to indicate the type of SDM you apply in developing the system in your organization.

1 To what extent is your organization/ company use the following standard system development methods at present? You may mark more than one	To a greater extent	To some extent	To a little extent	Not at all
1.1 water fall	1	2	3	4
1.2. extreme programming (XP)	1	2	3	4
1.3. Rational unified Process (RUP)	1	2	3	4
1.4. James Martins RAD (JMRAD)	1	2	3	4
1.5. structured Analysis, design and implementation of information (STRADIS)	1	2	3	4

2 which of the above standard systems development methods were change fit the specific needs of your organization and how significant were the changes. You may mark more than	Did not change	Minor change	Major change
2.1 water fall	1	2	3
2.2. extreme programming (XP)	1	2	3
3.3. Rational unified Process (RUP)	1	2	3
4.4. James Martins RAD (JMRAD)			

5.5. structured Analysis, design and implementation of information (STRADIS) 5.6. other please specify	1 1	2 2	3 3

3 For how long your system development methodology been used in your company			
3.1 less than a year	1		
3.2. 1 – 5 years	2		
3.3. 6 – 10	3		
3.4. 11 years or more	4		
3.5. don't know	5		

Return deadline:

APPENDIX 2: Ethical clearance

RESEARCH AND INNOVATION
OFFICE OF THE DIRECTOR

NAME OF RESEARCHER/INVESTIGATOR:

Mr B Macheque

Student No:
11612881

PROJECT TITLE: **Unbounded rule-based expert system for selecting software development methodologies.**

PROJECT NO: **SMS/18/BIS/04/0905**

SUPERVISORS/ CO-RESEARCHERS/ CO-INVESTIGATORS

NAME	INSTITUTION & DEPARTMENT	ROLE
Prof A Kadyamatimba	University of Venda	Supervisor
Prof NM Ochara	University of Venda	Co - Supervisor
Mr D Tutani	University of Venda	Co - Supervisor
Mr B Macheque	University of Venda	Investigator – Student

ISSUED BY:
UNIVERSITY OF VENDA, RESEARCH ETHICS COMMITTEE

Date Considered: May 2018
Decision by Ethical Clearance Committee Granted
Signature of Chairperson of the Committee: .....
Name of the Chairperson of the Committee: Senior Prof. G.E. Ekosse

UNIVERSITY OF VENDA
DIRECTOR
RESEARCH AND INNOVATION
2018 -05- 10
Private Bag X5050
Thohoyandou 0950

APPENDIX 3: Prof Reading

Office No. C7-4
Department of English
University of Venda
P/Bag X 5050
Thohoyandou
0950
15 February 2019

Dear Sir/Madam

This serves to confirm that I proof-read and edited dissertation entitled "Unbounded Rule-Based Expert System for Selecting Software Development Methodologies" by Macheque Vhutshilo, student number: 11612881.

I have suggested a few amendments, provided the changes I recommended are effected to the text, the language is of an acceptable standard.

Please don't hesitate to contact me for any enquiry.

Regards

Dr. Hlaviso Motlhaka
English Lecturer
Department of English
University of Venda
Tel: 015 962-8185
079-721-0620
078 196-4459
E-mail: *hlaviso.motlhaka@univen.ac.za*
Website: Website:http://www.univen.ac.za/

University of Venda
A quality driven, financially sustainable, rural-based comprehensive university.

APPENDIX 4**: Unbounded Rule-Based Expert System(URB-ES) coding**

SECTION A: URB-ES code flow

Declaring the variables required for a successful database and server connection

```
define("server","localhost");
define("user", "root");
define("password","");
define("database", "sdm");


$conn = mysqli_connect(server, user, password, database);
if($conn){
        //echo "connection success!";
}else{
        //echo "connection fail";
}
?>
```

2. Registering a user

```
<?php
$data = array();
require 'index.php'; //adding the index.php for connection and quering the database tables


if ( !isset($_POST['fname']) OR !isset($_POST['email']) OR !isset($_POST['password']) ){
        echo "Failure!";
}
else{
        $fullname = $_POST['fname'];
        $email = $_POST['email'];
        $password = $_POST['password'];


        $s = "INSERT INTO tbl_user(user_fullname, user_email, user_password)
VALUES('$fullname', '$email', '$password')";
```

```
        $res = mysqli_query($conn, $s);

        if( $res ){
            echo "User registered!";
        }
        else{
            echo "User failed to register!";
        }
}
```

1. declaring the variables required for a successful database and server connection

```
define("server","localhost");
define("user", "root");
define("password","");
define("database", "sdm");

$conn = mysqli_connect(server, user, password, database);

if($conn){
        //echo "connection success!";
}else{
        //echo "connection fail";
}
```

4. This is the code for registering a user

```
package assignment2.sdm;

import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
```

```
import assignment2.sdm.CustomObjects.Constants;
import assignment2.sdm.CustomObjects.User;

public class RegisterActivity extends AppCompatActivity {

    private EditText etFullName, etEmail, etPassword;
    private Button mButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        etFullName = (EditText) findViewById(R.id.etFNames);
        etEmail = (EditText) findViewById(R.id.etEmail);
        etPassword = (EditText) findViewById(R.id.etPassword);

        mButton = (Button) findViewById(R.id.btnRegister);
        register();
    }

    private void register(){
        final String fullname = etFullName.getText().toString(), email =
etEmail.getText().toString(), password = etPassword.getText().toString();
        mButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                new Register(new User(fullname, email, password)).execute();
            }
        });
    }


    class Register extends AsyncTask<Void,Void,Boolean>{

        private User user;

        public Register(User user) {
            super();
            this.user = user;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected Boolean doInBackground(Void... voids) {

            URL url;
            HttpURLConnection connection;
            try{

                url = new URL(Constants.reg_url);
                connection = (HttpURLConnection) url.openConnection();
                connection.setConnectTimeout(10*1000);
                connection.setRequestMethod("POST");
                connection.setDoInput(true);
                connection.setDoOutput(true);

                StringBuilder builder = new StringBuilder();
                String line;
                String data = "";

                OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
                data = URLEncoder.encode("fullname", "UTF-
```

```
8")+"="+URLEncoder.encode(user.fname,"UTF-8")+"&"+
                        URLEncoder.encode("email", "UTF-
8")+"="+URLEncoder.encode(user.email,"UTF-8")+"&"+
                        URLEncoder.encode("password", "UTF-
8")+"="+URLEncoder.encode(user.password,"UTF-8");
                writer.write(data);

            }catch (Exception e){
                Log.e("ASYNC", e.getMessage());
            }

            return null;
        }

        @Override
        protected void onPostExecute(Boolean aBoolean) {
            super.onPostExecute(aBoolean);
            startActivity(new Intent(RegisterActivity.this, MainActivity.class));
            finish();
        }
    }

}
```

5. Code for Main Activity that provides the ability to choose

```
package assignment2.sdm;

import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

import assignment2.sdm.CustomObjects.Constants;
import assignment2.sdm.CustomObjects.User;

public class RegisterActivity extends AppCompatActivity {

    private EditText etFullName, etEmail, etPassword;
    private Button mButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        etFullName = (EditText) findViewById(R.id.etFNames);
        etEmail = (EditText) findViewById(R.id.etEmail);
        etPassword = (EditText) findViewById(R.id.etPassword);

        mButton = (Button) findViewById(R.id.btnRegister);
        register();
    }

    private void register(){
```

```
        final String fullname = etFullName.getText().toString(), email =
etEmail.getText().toString(), password = etPassword.getText().toString();
        mButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                new Register(new User(fullname, email, password)).execute();
            }
        });
    }


    class Register extends AsyncTask<Void,Void,Boolean>{

        private User user;

        public Register(User user) {
            super();
            this.user = user;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected Boolean doInBackground(Void... voids) {

            URL url;
            HttpURLConnection connection;
            try{

                url = new URL(Constants.reg_url);
                connection = (HttpURLConnection) url.openConnection();
                connection.setConnectTimeout(10*1000);
                connection.setRequestMethod("POST");
                connection.setDoInput(true);
                connection.setDoOutput(true);

                StringBuilder builder = new StringBuilder();
                String line;
                String data = "";

                OutputStreamWriter writer = new
OutputStreamWriter(connection.getOutputStream());
                data = URLEncoder.encode("fullname", "UTF-
8")+"="+URLEncoder.encode(user.fname,"UTF-8")+"&"+
                        URLEncoder.encode("email", "UTF-
8")+"="+URLEncoder.encode(user.email,"UTF-8")+"&"+
                        URLEncoder.encode("password", "UTF-
8")+"="+URLEncoder.encode(user.password,"UTF-8");
                writer.write(data);

            }catch (Exception e){
                Log.e("ASYNC", e.getMessage());
            }

            return null;
        }

        @Override
        protected void onPostExecute(Boolean aBoolean) {
            super.onPostExecute(aBoolean);
            startActivity(new Intent(RegisterActivity.this, MainActivity.class));
            finish();
        }
    }
```

```
}
```

6. Consulting Activity Code

```
package assignment2.sdm;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.View;

import com.google.gson.Gson;

import assignment2.sdm.Adapters.CriteriaAdapter;
import assignment2.sdm.CustomObjects.CriteriaList;
import assignment2.sdm.CustomObjects.Utilities;

public class ConsultActivity extends AppCompatActivity {

    public CriteriaList list;
    public CriteriaList mList;
    public RecyclerView rvList;
    private FloatingActionButton fabCommit;

    //this adapter will contain the criteria array of all critic
    public CriteriaAdapter adapter;

    Gson gson; //manipulates data from JSON to GSon and to any kind of object
preferred

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_consult);
        list = Utilities.fillList();
        mList = new CriteriaList();
        rvList = (RecyclerView) findViewById(R.id.rv_items);
        fabCommit = (FloatingActionButton) findViewById(R.id.fabCommit);
        rvList.setLayoutManager(new LinearLayoutManager(this,
LinearLayoutManager.VERTICAL, false));
        adapter = new CriteriaAdapter(this, list);
        rvList.setAdapter( adapter );
        fabCommit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                gson = new Gson();
                String list_to_json = gson.toJson(mList);
                Intent intent = new Intent(ConsultActivity.this,
WeightActivity.class);
                intent.putExtra("list", list_to_json);
                startActivity(intent);
            }
        });
    }
}
```

7. Weighing Activity

```
package assignment2.sdm;

import android.content.Intent;
```

```
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutCompat;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.View;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.lang.reflect.Type;

import assignment2.sdm.Adapters.WeightAdapter;
import assignment2.sdm.CustomObjects.Criteria;
import assignment2.sdm.CustomObjects.CriteriaList;
import assignment2.sdm.CustomObjects.ResultsModelsActivity;
import assignment2.sdm.CustomObjects.Weight;
import assignment2.sdm.CustomObjects.Weights;

public class WeightActivity extends AppCompatActivity implements
View.OnClickListener{

    private CriteriaList list;
    private Gson gson;
    private String EXTRA_KEY_NAME = "list";
    private String json;

    private RecyclerView rvWeights;
    private FloatingActionButton fabContinue;
    private LinearLayoutManager manager;
    private WeightAdapter adapter;
    private Weights weights;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_weight);

        rvWeights = (RecyclerView) findViewById(R.id.rvWeights);
        fabContinue = (FloatingActionButton) findViewById(R.id.fabContinue);
        manager = new LinearLayoutManager(this, LinearLayoutManager.VERTICAL,
false);
        rvWeights.setLayoutManager(manager);

        try{
            json = getIntent().getStringExtra(EXTRA_KEY_NAME);
            gson = new Gson();
            Type typeCriteriaList = new TypeToken<CriteriaList>(){}.getType();
            list = gson.fromJson(json, typeCriteriaList);

            if (list == null && list.size() < 0){
                Log.e("List", "Nothing is selected!");
            }else{
                Log.e("List", "Existing, with "+list.size()+" length!");
                adapter = new WeightAdapter(this, extractWeights());
                rvWeights.setAdapter(adapter);
            }
            fabContinue.setOnClickListener(this);

        }catch (Exception e){
            Log.e("Error", e.getMessage());
        }
    }

    @Override
    public void onClick(View view) {
```

```
        switch (view.getId()){
            case R.id.fabContinue:

                startActivity(new Intent(this, ModelACtivity.class));

                break;
        }
    }

    Weights extractWeights(){
        weights = new Weights();
        for (Criteria criteria: list){
            weights.add(new Weight(criteria));
        }

        return weights;
    }

}
```

8. Model Activity

```
package assignment2.sdm;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;

import assignment2.sdm.Adapters.Model;
import assignment2.sdm.Adapters.ModelAdapter;
import assignment2.sdm.CustomObjects.Models;

public class ModelACtivity extends AppCompatActivity {

    Models models;
    RecyclerView rcModels;
    ModelAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_model_activity);

        fill();
        adapter = new ModelAdapter(this, models);
        rcModels = (RecyclerView) findViewById(R.id.rvModels);

        rcModels.setLayoutManager(new LinearLayoutManager(this,
LinearLayoutManager.VERTICAL, false));
        rcModels.setAdapter(adapter);
    }
    void fill(){
        models = new Models();
        models.add(new Model("Waterfall Model"));
        models.add(new Model("Extreme Programming"));
        models.add(new Model("Spiral Model"));
        models.add(new Model("Reuse Oriented"));
        models.add(new Model("SCRUM"));
        models.add(new Model("Crystal"));
        models.add(new Model("Dynamic"));
        models.add(new Model("Agile"));
        models.add(new Model("Feature-Driven"));
        models.add(new Model("Lean"));
        models.add(new Model("Joint Application"));
    }
}
```