An investigation of the suitability of agile system development methodologies for the development of data warehouses

J Grey Hons BSc

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

Supervisor: Prof HM Huisman
Co-supervisor: Dr R Goede

November 2006

ABSTRACT

The aim of this study is to investigate whether agile system development methodologies (ASDMs) are suitable for the development of data warehouses. To reach this aim, a literature study was conducted on the relatively settled ASDMs by firstly defining a system development methodology (SDM) and an ASDM. Each ASDM explanation contains the identified key factors, unique process model, and method of use. The seven ASDMs investigated in this study, are: Dynamic System Development Methodology (DSDM), Scrum, Extreme Programming (XP), Feature Driven Development (FDD), Crystal ASDMs - especially Crystal Clear (CC), Adaptive Software Development (ASD), and Lean Development (LD).

In addition, a literature study is conducted on the data warehouse approaches of Inmon (1996) and Kimball *et al.* (1998). Each data warehouse approach is explained using the architecture, lifecycle and four distinct phases within the lifecycle. The four distinct phases include: collecting requirements, data modelling, data staging, and data access and deployment. After this was done, Inmon and Kimball's approaches were compared.

After studying the ASDMs and data warehousing approaches, theoretical deductions were made regarding the suitability of ASDMs in data warehouse development. General deductions (including the applicability of agile processes) for all ASDMs as well as unique deductions for each of the seven ASDMs mentioned above were formulated in theory. The theoretical deductions lead to the limitation of the empirical section of the study to the suitability of ASDMs within the framework of Kimball's approach.

Theoretical deductions were empirically tested by conducting an interpretive experiment where seven data warehouse development teams used an assigned ASDM to develop a data warehouse. The data warehouse consisted of one data mart. Each team was expected to develop their data mart

incrementally, one sub-data mart at a time. Every sub-data mart was developed iteratively to form the data mart. The data mart was then deployed as a whole (including everything from collecting requirements to report generation) to the users.

The findings of the study are a combination of the theoretical deductions and interpretive results (propositions) of the interpretive experiment conducted. These findings indicate that ASDMs are suitable to develop data warehouses in a constantly changing environment.

OPSOMMING

Die hoofdoel van hierdie studie was om te bepaal of ASDMs (vinnig aanpasbare stelselontwikkelingsmetodologieë) toepaslik is vir die ontwikkeling van 'n datapakhuis. Om dit te kon vasstel, is 'n literatuurstudie gedoen in 'n poging om 'n SDM (stelselontwikkelingsmetodologie) en ASDMs te definieer. Dit sluit 'n beskrywing van elke ASDM se sleutelidentifiseringsfaktore, unieke prosesmodel en metode van gebruik in. Die sewe ASDMs wat in hierdie studie bestudeer is, is: Dynamic System Development Methodology (DSDM), Scrum, Extreme Programming (XP), Feature Driven Development (FDD), Crystal ASDMs – veral Crystal Clear (CC), Adaptive Software Development (ASD), en Lean Development (LD).

Tweedens is literatuur oor Inmon (1996) en Kimball *et al.* (1998) se datapakhuisontwikkelingbenaderings bestudeer. Elke datapakhuisbenadering se boustyl (*architecture*) komponente, lewenssiklus (*lifecycle*) en vier afsonderlike fases (in die lewenssiklus) is beskryf. Die vier fases wat beskryf is vir elke datapakhuisbenadering, is: behoeftebepaling, datamodellering, data-opstelling, en datatoegang en ontplooiing. Daarna, is Inmon (1996) en Kimball *et al.* (1998) se datapakhuis benaderings met mekaar vergelyk.

Na afloop van die teoretiese ondersoek na ASDMs en datapakhuisontwikkeling, is teoretiese afleidings gemaak rakende die geskiktheid van ASDMs binne datapakhuisontwikkeling. Algemene afleidings oor alle ASDMs, insluitend die toepaslikheid van vinnig aanpasbare (*agile)* prosesse, en spesifieke afleidings oor elkeen van die bogenoemde ASDMs, is uit die teorie geformuleer. Na aanleiding van hierdie teoretiese afleidings is die empiriese gedeelte van die studie beperk tot die geskiktheid van ASDMs in datapakhuisontwikkeling binne die raamwerk van Kimball *et al.* (1998) se datapakhuisbenadering.

Die teoretiese afleidings is empiries getoets deur 'n interpretatiewe eksperiment uit te voer, waar sewe datapakhuisspanne 'n toegekende ASDM gebruik het

om 'n datapakhuis te ontwikkel. Die datapakhuis het slegs uit een "data mart" bestaan. Van elke span is verwag om 'n "data mart" inkrementeel te ontwikkel, een "sub-data mart" op 'n keer. Elke "sub-data mart" is iteratief ontwikkel om uiteindelik in geheel 'n "data mart" te vorm. Die "data mart" is dan as 'n geheel (van behoeftebepaling tot verslag generasie) aan die gebruikers ontplooi.

Die bevindings van die studie is 'n kombinasie van die teoretiese afleidings en die interpretatiewe resultate (proposisies) van die interpretatiewe eksperiment. Die bevindinge toon dat ASDMs geskik is om datapakhuise in 'n konstante veranderende omgewing te ontwikkel.

ACKNOWLEDGEMENTS

A S N F

"A Son Never Forgets"

I want to give thanks to my father and mother who taught me to be the man that I am today. I want to thank them for their love and support and for the example they set as parents.

I want to thank my fiancée for her encouraging words and ongoing support during the past year.

A special thank you goes out to Prof Magda Huisman, for her insight and guidance in the work that I have been doing, and for always assuring me that I have done good work.

I would also like to thank Dr Roelien Goede for the role that she fulfilled as co-sponsor and Thalyta Swanepoel for revising grammar and spelling.

Most importantly, I want to give praise to the Lord Jesus Christ for blessing me with wonderful people in my life and for giving me the opportunity to use the talents that He has given me.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

CHAPTER 2

AGILE SYSTEM DEVELOPMENT METHODOLOGIES (ASDMs)

CHAPTER 3

DATA WAREHOUSING

CHAPTER 4

THEORETICAL DEDUCTIONS: SUITABILITY OF ASDMs FOR DATA WAREHOUSE DEVELOPMENT

CHAPTER 1

INTRODUCTION

1.1 Proposed title

An investigation of the suitability of agile system development methodologies for the development of data warehouses

1.2 Key words

System development methodology (SDM); agile system development methodology (ASDM); data warehouse; business intelligence (BI); Dynamic System Development Methodology (DSDM); Scrum; Extreme Programming (XP); Feature Driven Development (FDD); Crystal ASDMs (specifically Crystal Clear (CC)), Adaptive Software Development (ASD); Lean Development (LD)

1.3 Background and problem statement

Information technology projects tend to change due to elements of uncertainty such as constant changing requirements, project time and budget instability, intelligence and the team's ability to respond to new demands (Chin, 2003)[1]. As a result of the evolving business environment, the requirements set by business users, change. Consequently, there will be a demand for SDMs with the ability to adapt to a changing environment. SDMs are collections of phrases, procedures, rules, techniques, tools, documentation, management and training used to develop information systems (Avison and Fitzgerald, 2003:80)[2].The primary objective of using ASDMs in organisations is to deliver information systems quickly, change quickly and to change as frequently as possible

[1] References containing no page numbers are website publications (normally in html or .pdf format) that contain no page numbers.

[2] References containing page numbers are used for referencing full text internet articles, journal articles, newspaper articles, magazine articles and books.

(Highsmith, 2002b).

Iivari and Maansaari (1998:502) classify conceptual problems related to the use of the term SDM into two types of inconsistency namely; scope and category. Avison and Fitzgerald (2003:528) state that an SDM is more than just a method; it has certain characteristics that emphasises the inclusion of a philosophical view. Therefore, according to Huisman and Iivari (2006:32) an SDM can be defined as a combination of the following:

- *A system development approach* is the philosophical view on which a methodology is based. It is the set of goals, fundamental concepts, guiding principles and beliefs of the system development process that drive interpretations and actions in system development (Iivari *et al.*, 1998:165-166; Iivari *et al.*, 1999:2). Examples of system development approaches include the process-oriented approach, object-oriented approach, and information modelling.
- *A system development process model:* Wynekoop and Russo (1993:182) define a process model as a representation of the sequences of stages through which a system evolves. Examples of process models are the waterfall model, linear lifecycle, and the spiral model.
- *A system development method,* according to Brinkkemper (1996:275), is "an approach to perform a system development project, based on a specific way of thinking, consisting of definitions and rules, structured in a systematic way in development activities with corresponding development products". He also states that it is a "way of investigation". Wynekoop and Russo (1993:182) describes a method as a systematic approach to conduct at least one complete phase of system development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of system development and the target system. Examples include Information Engineering, Structured Systems Analysis and Design Method (SSADM) and Jackson System Development.
- *A system development technique:* Techniques can be described as a

procedure, possibly with a prescribed notation, to perform a development activity (Brinkkemper, 1996:276). Examples include entity relationship diagrams (ERD), decision tables, and data flow diagrams.

This study will investigate seven relatively settled new ASDMs. These respective methodologies are, in the order that they were developed by system designers and developers; Dynamic System Development Methodology [1994], Scrum [1995], Extreme Programming [1998], Feature Driven Development [1998], Crystal ASDMs (specifically Crystal Clear [1999]), Adaptive Software Development [2000] and the most recent Lean Development [from 2000], which started as Lean Manufacturing. Lean Manufacturing was used in 1980 by Japanese automobile companies (Honda and Toyota), to compete with American automobile companies.

It appears ASDMs are recent developments and practitioners (other than the authors of the ASDMs) do not specifically know in what environments and circumstances it will function successfully. For example, Extreme Programming (XP) may work for a project tested by its author in a specific environment, while in some organisations XP is partially adopted in projects (Aveling, 2004:94). According to Abrahamsson *et al.* (2002:26) XP is growing, Crystal ASDMs uses only the two methods for the smallest teams (Control Chaos, 2006) and Scrum, which has the ability to integrate with XP, is gaining popularity. Dynamic System Development Methodology (DSDM), on the other hand, is widely used in the United Kingdom. Adaptive Software Development (ASD) seems to have no research reported in literature. Feature Driven Development (FDD) is still evolving (Abrahamsson *et al.,* 2002:55). Lean Development (LD), which was only documented recently, receives growing attention in product development, but not much is published about the success of using LD in a project.

Data warehousing is another relatively new field in Computer Science and project development. Bill Inmon, the father of data warehousing (Inmon, 1996), and the dimensional data warehousing expert Ralph Kimball (Kimball *et al.*,

1998), have different approaches and architectures concerning the development of a data warehouse. Inmon uses the hub & spoke architecture, while Kimball prefers high-level technical architecture. There is a great difference in implementation between Inmon (1996) and Kimball's (1998) architecture's when examining the five roles of a data store namely, intake, integration, distribution, access, and delivery (The Data Warehousing Institute, 2004:3-7). These approaches will be investigated to determine which has the potential to develop a data warehouse using ASDMs. In order to determine which data warehouse approach has the potential and characteristics to develop a data warehouse using ASDMs, theoretical deductions will be explained. The explained theoretical deductions will bring data warehousing and ASDMs together by evaluating the suitable and unsuitable characteristics of every ASDM for the different phases of data warehouse development of Inmon and Kimball's approaches. After the evaluation the most suitable data warehousing approach will be chosen to develop seven data warehouses using the seven different ASDMs (in team formation).

There is little evidence showing that an ASDM has the ability to be used in the development of data warehouses. ASDMs (particularly FDD and XP) do, however, have characteristics that could contribute to the successful development of data warehouses (Graziano, 2005). The data warehouse development lifecycle and phases also show opportunity for ASDMs to be successful. Graziano (2005) furthermore argues that data warehouses could be developed using the twelve specific principles of the Agile Manifesto.

The Agile Alliance is "a non-profit organisation that supports individuals and organisations that use agile approaches to develop software" (Agile Alliance, 2006). The Agile Alliance use the priorities in the Manifesto for Agile Software Development to deliver a product of value and of high quality faster to users and organisations. The Manifesto for Agile Software Development consists of four values and twelve principles on which the seventeen founding members agree upon. The seventeen founding members agree that there are better ways

discovered for developing software. While investigating these ways and helping others to implement them, they have come to value (Fowler & Highsmith, 2001):

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

The seventeen founding members of the Manifesto for Agile Software Development follow the following twelve principles (Fowler & Highsmith, 2001):

- "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity - the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective,

then tunes and adjusts its behaviour accordingly."

The seventeen founding members of the Manifesto for Agile Software Development, include: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas.

Against the above background the main research question is as follows:

Can ASDMs be used in the development of data warehouses?

1.4 Reasons for the study

ASDMs focus on incremental development, people and user requirement satisfaction during system development to deliver a product that is up to date in a constant changing environment.

Changing SDMs

Avison and Fitzgerald (2003:79-82) refer to four era's in system development to explain the constant changing environment:

The pre-methodology era: Early computer systems (1960 – 1970) were designed without formal methodologies (Avison and Fitzgerald, 2003:79). User requirements were rarely well defined and this resulted in user as well as business discontentment.

The early methodology era: Computer-based applications were developed by identifying phases that would improve the management of system development to introduce a model commonly known as the waterfall model. According to the waterfall model, a new phase can only start once the previous phase has been completed. The unstable and inflexible computer systems did however fail to

meet business needs during this era (Avison & Fitzgerald, 2003:79).

The methodology era: Methodologies were introduced with the aim to assist computer systems to move beyond these above mentioned limitations. The methodologies that appeared during this era can be classified as structured, data oriented, prototypical, object oriented, participative, strategic or systematic (Avison and Fitzgerald, 2003:80).

The post-methodology era: In the late 1990's researchers started questioning the worth of concepts used in earlier methodologies. Consequently some methodologies were abandoned, others adapted and new methodologies were used in organisations (Avison & Fitzgerald, 2003:80). The reason for studying ASDMs is because they are part of the most recent era of SDMs.

Studying new and relatively settled ASDMs

ASDMs are part of the "post-methodology era" that started in the late 1990's. In this study the researcher will be unable to explain all ASDMs. The reason for choosing these seven ASDMs are because they are new but relative settled ASDMs in practice. According to Highsmith (2002a:6) these ASDMs are the core group, as set out in the Agile Software Development Manifesto (Highsmith 2002a:6; Lindstrom & Jeffries, 2004:44). Although Lindvall *et al.* (2002:199) mentioned six popular ASDMs, this study will focus on all, expect Agile Modelling. Furthermore, ASD and LD will be included in this study. Another reason for only choosing these seven ASDMs is because they were developed in a sequence starting in 1994 (DSDM) and ending with the most recent ASDM, LD.

Changing environment and requirements

According to Lindvall *et al.* (2002:197) some practitioners in the mid-1990's found requirements documentation and design development steps frustrating,

and in some circumstances even impossible. The plan-driven methods, like the waterfall model and iterative approaches may show difficulties when change is expected (Boehm, 2002:69).

Technology is an ever changing reality of which practitioners and developers must take notice. Customers have become unable to explain their definite requirements and needs because of the changing requirements (Lindvall *et al.*, 2002:198). Lindvall *et al.* (2002:198) states that as a result, consultants developed methodologies and practices to "embrace and respond to inevitable change they were experiencing". Today, these methodologies according to Lindvall *et al.* (2002:198) are known as ASDMs with characteristics of incremental development and the ability to adapt to change. Thus, in an ever changing environment with shifting requirements, ASDMs allow teams to adapt quickly.

New methodologies applied to data warehousing

In Kent Graziano's (2005) opinion, ASDMs can be used in the development of data warehouses. Referring to principle eight of the agile manifesto (Agile processes promote sustainable development and the sponsors, developers and users should be able to maintain a constant pace indefinitely), Graziano (2005) argues that a standard repeatable ASDM should be used.

According to Graziano (2005) FDD seems to be most applicable to data warehouses that can use a feature as one data mart report, and a feature set as a star-schema data mart or a data warehouse subject area. The primary goal of FDD is to deliver the model in smaller components (increments).

Team huddles, a method used in FDD and Scrum, is effective. This method entails a daily meeting (stand-up meetings) to discuss problems and set new objectives (Graziano, 2005). It also motivates the group and improves team work. XP, on the other hand, uses pair programming which results in a faster,

more accurate ETL (extract, transform and load) programming. Pair programming involves two programmers programming on one PC.

Graziano chose these two ASDMs as candidates to develop his data warehouse based upon his knowledge of data warehousing and ASDMs. In this study however the seven chosen ASDMs will be tested in an interpretive experiment where seven data warehouses will be developed to determine whether ASDMs can be applied to data warehouse development. The interpretive experiment will be conducted using seven teams that will use a randomly assigned ASDM to develop a data warehouse in a changing environment. It will then be investigated whether some of the theoretical deduction made is confirmed and whether additional information could be attained from the data warehouse development processes.

1.5 Research aims and objectives

The main research question of this study is to investigate the suitability of ASDMs for the development of data warehouses. In order to investigate this research question, the following research objectives/aims will be addressed:

- Investigate the suitability that ASDMs can be used in data warehouse development.
- Study seven new and relatively settled ASDMs.
- Study Inmon (1996) and Kimball's (1998) approaches towards data warehouse development.
- Evaluate the suitable and unsuitable characteristics of every ASDM for the different phases of data warehouse development of Inmon and Kimball's approaches.
- Conduct an interpretive experiment to test whether ASDMs can be applied to data warehouse development, by using seven teams, where every team has to use a randomly assigned ASDM to develop a data warehouse in a changing environment.

- Investigate whether the theoretical deductions made was confirmed by the interpretive experiment.
- Combine the confirmed theoretical deductions and interpretive results to present findings on the suitability of ASDMs towards data warehouse development.

1.6 Research approach

The research is conducted in the interpretive research paradigm as described by Lee (1999:17). "In contrast to the world of positivism, the world of interpretativism gives explicit recognition to the 'life world'. Not originating in the natural sciences, interpretativism involves research procedures such as those associated with ethnography (from anthropology), participant observation (from sociology), history, and hermeneutics, all of which give explicit recognition to the world of consciousness and humanly created meanings. In most interpretative approaches, a central idea is 'mutual understanding' – the phenomenon of a person understanding (i.e. 'interpreting') what another person means – whether it is a person engaged in everyday life taking a natural attitude to understanding another person in everyday life, or it is a person engaged in scientific research taking a calculated scientific attitude to understanding everyday people in their everyday lives."

An interpretative experiment is conducted with cross-case analysis (as described by Seaman, 1999:567-569) as data analysis method. The researcher will study seven relatively settled ASDMs as well as the different data warehouse approaches of Inmon (1996) and Kimball *et al.* (1998) in order to establish whether ASDMs can be used to develop data warehouses in a constant changing environment. The suitable and unsuitable characteristics of every ASDM will be explained (using deductions) in the different phases of data warehouse development for Inmon and Kimball's approaches. A suitable data warehousing approach will be decided upon based on these deductions. The deductions will be tested by conducting an interpretive experiment were seven

teams will develop data warehouses using the seven chosen ASDMs. After the interpretive experiment is conducted, cross-case analysis will be used to analyse the findings of the seven data warehouses that was developed with the seven different ASDMs. Thus, propositions will be listed that is applicable to all the ASDM data warehousing projects using cross-case analysis.

The unique individual characteristics for every ASDM will be explained as theoretical deductions after cross-case analysis is completed. Lastly, new findings will be explained by combining the theoretical deductions with the results of the interpretive experiment (propositions) for all ASDMs, as well as every individual ASDM.

1.7 Chapter Outline

- Chapter 1: *Introduction.*
- Chapter 2: *Agile System Development Methodologies (ASDMs)*: In this chapter a SDM will be defined as well as an ASDM. Secondly, the new and relatively settled seven ASDMs used in practice will be explained. Lastly the effectiveness of ASDMs used today will be described.
- Chapter 3: *Data Warehousing*: In chapter 3 the term BI as well as the BI framework will be discussed. Secondly, a data warehouse and definitions associated with data warehousing are defined. Thirdly, Inmon (1996) and Kimball's (1998) approaches will be discussed using their architectures and lifecycles. Both lifecycles will be explained using four phases that includes, collecting requirements, data modelling, data staging, and data access and deployment. Lastly, Inmon and Kimball's approaches will be compared.
- Chapter 4: *Theoretical deductions*: *Suitability of ASDMs for data warehouse development*. In this chapter, the suitability of the use of ASDMs in data warehousing will be investigated from a theoretical point of view. The general findings associated with the characteristics of all ASDMs in data warehouse development will be explained, including the applicability of the nine core values of all agile process. Next, the explanation of each ASDM's

suitability towards data warehouse development will follow. The theoretical deductions will be explained under each ASDM for Kimball *et al.* (1998) and Inmon's (1996) approaches.

- Chapter 5: *Application of ASDMs for data warehouse development.* In chapter 5 the theoretical deductions made in chapter 4 will be interpretively tested. In this chapter it will be explained how the interpretive experiment was designed, how the data was collected, and how the data was analysed (using cross-case analysis) for the seven different development teams. Seven teams of equal strength developed a data warehouse using Kimball's data warehouse development lifecycle. Each team used their assigned ASDM to guide their activities during the data warehouse development process. Furthermore, it will be explained how the interpretive experiment was conducted. Lastly, the researcher will determine whether the data warehousing projects of the seven teams was successful.
- Chapter 6: *Confirmed Findings*: After examining the theoretical deductions made in chapter 4 and the interpretative experiment results in chapter 5, these deductions and interpretive results (propositions) will be combined in chapter 6. This will be done by presenting findings where certain ASDM areas, steps, properties or principles can be applied to the data warehouse development phases, based on the experience gained from chapter 4 and chapter 5.

1.8 Limitations

- *Size of the team*: ASDMs need a team to function efficiently during development. The team for this study will only consist of three to four members. Some ASDMs are only proven to work for large projects. Therefore a small team may limit the interpretive experiment.
- *Software*: Finding software that enables the researcher to use a specific ASDM for the development of a data warehouse can be a timely process.
- *Tools*: The use of unreliable tools may result in losing important data during

the cleaning process. The researcher will either write a programme, or use reliable tools to manage and clean the data.

CHAPTER 2

AGILE SYSTEM DEVELOPMENT METHODOLOGIES (ASDMs)

2.1 Introduction

In this chapter the definition of a system development methodology (SDM) will be investigated since no universally accepted definition exists. Secondly, an agile system development methodology (ASDM) will be described, and what it means for an organisation to be agile will be discussed. Furthermore, the seven core set ASDMs, which are commonly used in practice, will be explained. These are:

- Dynamic Systems Development Methodology (DSDM)
- Scrum
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Crystal ASDMs
- Adaptive Software Development (ASD)
- Lean Development (LD)

The explanation of every ASDM will contain the identifying key factors as well as its unique process model and method of use. Lastly, the effectiveness of ASDMs at implementation level will be discussed using papers and surveys conducted by experts in agile project development. Organisations have a growing interest in ASDMs because of their adaptive behaviour, which holds that they have the ability to adapt to change frequently, and speedily.

2.2 Definition of a system development methodology (SDM)

Trying to define an SDM is a difficult task. There is no universally accepted, concise definition of information SDM (Avison & Fitzgerald, 2003:527;

Wynekoop & Russo, 1997:48; Iivari *et al.*, 1999:1).

The first problem trying to define an SDM is the "method versus methodology" debate. Researchers have different views. Some argue that the term "methodology" has no place in information systems, because it literally means a "science of methods" (Schach, 1997:23), while others argue that the terms can be applied interchangeably (Hardy *et al.*, 1995:467-468; Saeki, 1998:925). Others argue that methodologies encompass methods or that methods encompass methodologies (Palvia & Nosek, 1993:73).

There are conceptual problems related to the use of the term "system development method/methodology". Iivari and Maansaari (1998:502-503) classify these problems as "scope problems" and "category problems". Scope problems include instances where a system development method/methodology covers the system's development process, or where there is concern about the aspects that should cover the system development method/methodology. Category problems include difficulty distinguishing between techniques and system development methods/methodologies.

Despite category problems and scope problems, four elements can be identified in the various definitions of a system development method/methodology (Huisman & Iivari, 2006:32):

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

Avison and Fitzgerald (2003:20,527) argue that the term "methodology" is a much wider concept than the term "method", because a methodology has certain characteristics that are not implied by method, and a methodology

includes a philosophical view. In this study, the researcher uses the term "methodology" to cover all four elements, because it's a much larger concept than the term "method", and does not aim to contribute to the method versus methodology debate. Therefore the term "system development methodology" (SDM) will be used, instead of "system development method".

Consequently, an SDM can be defined as a combination of the following (Huisman & Iivari, 2006:32):

- *A system development approach* is the philosophical view on which a methodology is based. It is the set of goals, fundamental concepts, guiding principles and beliefs of the system development process that drive interpretations and actions in system development (Iivari *et al.*, 1998:165-166; Iivari *et al.*, 1999:2). Examples of system development approaches include the process-oriented approach, object-oriented approach, and information modelling.
- *A system development process model:* Wynekoop and Russo (1993:182) define a process model as a representation of the sequences of stages through which a system evolves. Examples of process models are the waterfall model, linear lifecycle, and the spiral model.
- *A system development method,* according to Brinkkemper (1996:275), is "an approach to perform a system development project, based on a specific way of thinking, consisting of definitions and rules, structured in a systematic way in development activities with corresponding development products". He also states that it is a "way of investigation". Wynekoop and Russo (1993:182) describes a method as a systematic approach to conduct at least one complete phase of system development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of system development and the target system. Examples include Information Engineering, Structured Systems Analysis and Design Method (SSADM) and Jackson Systems Development.
- *A system development technique:* Techniques can be described as a

procedure, possibly with a prescribed notation, to perform a development activity (Brinkkemper, 1996:276). Examples include entity relationship diagrams, decision tables, and data flow diagrams.

By understanding the building blocks of an SDM, an agile system development methodology (ASDM) can now be defined.

2.3 Definition of an agile system development methodology (ASDM)

The primary goal of any ASDM is to make an organisation agile, in other words, giving the organisation the ability to adapt to change. However, the question rises: what does it mean to be agile? Highsmith (2002b) states that it means being able to deliver quickly, change quickly, and to change as often as necessary. Cockburn and Highsmith (2001a:120) explain what is new about ASDMs is not the practices they use, but their recognition of people as the primary drivers of project success, coupled with a primary focus on manoeuvrability, change and effectiveness. Fowler (2006) calls ASDMs the "new methodologies" because, due to their adaptive nature and people-first orientation, they have blossomed during the past 10 years.

Conboy and Fitzgerald (2004:108) explain that agility consists of two components namely flexibility and speed. Terms such as "fast", "rapid", "speed", and "quick" are commonly found in definitions of agility, thus for an organisation to practice agility, it must be able to respond "speedily and flexibly". An ASDM is more "adaptive than predictive", more "people-oriented than process-oriented" (Fowler, 2001).

The principles and guidelines of ASDMs are continuously being described and defined; and some of these principles include (Mendonca, 2002:505):

- "Agile processes that continuously respond to changes in the environment

- Appropriate selection of process components that reflect efficiency in addition to effectiveness
- An adaptive approach (frameworks) rather than adherence to predefined process rules
- Frequent, rapid delivery of smaller software components to achieve faster feedback
- A collaborative approach to development
- An expectation of change during the development process
- Outcomes are emergent, rather than fixed
- Creativity in problem solving
- Dynamic re-prioritization"

ASDM share common characteristics, including communication, incremental development and people. Their practices and emphases do vary, but the goal of all ASDMs is to make the organisation agile. An agile project can identify and respond to changes more quickly than a project following a more traditional approach (Cohen *et al.*, 2003).

According to Hislop *et al.* (2002:177) and DSDM Consortium (2005), there are nine principles that reflect the common core values of all agile processes:

- "Users must be actively involved throughout the development process
- Teams (including both users and developers) must be empowered to make decisions without explicit approval from higher management
- Frequent delivery of products has highest priority
- Deliverables are evaluated primarily with respect to their fitness for business purposes
- Rapid iterations and incremental delivery are key to converging on acceptable business solutions
- No changes are irreversible – backtracking to or reconstructing previous versions must be possible
- High-level requirements are frozen early to allow for detailed investigation of

their consequences

- Testing is integrated throughout the development live cycle
- Collaboration and cooperation among all stakeholders is the key to success"

Focusing on communications means project teams can make decisions and act on them immediately, rather than wait for correspondence. Development in iterations allows the team to quickly adapt to the changing environment and requirements.

According to Lindvall *et al.* (2002:201), ASDMs are:

- *Iterative:* A full system is delivered at the beginning of the project, before changes on each sub-system is done. A sub-system is released after its functionality has been changed because of new requirements.
- *Incremental:* The system is delivered in pieces. This means that the requirements are partitioned into small subsystems where after the new requirements and functionality is added.
- *Self-organizing*: The team is obliged to manage and organise themselves in order to complete the system within budget and time constraints.
- *Emergent:* Technology and requirements are allowed to emerge throughout the product development cycle. ASDMs have the ability to adapt to change so that new requirements can emerge and be implemented.

Organisations are increasingly adopting ASDMs within their projects (Good, 2003:28;). The effectiveness of ASDMs is still being studied, but the functionality in certain circumstances of most has been proven, and deserves system developers' attention (Mendonca, 2002:505; Ambler, 2002:9).

2.4 The seven ASDMs

In this segment of the study only the seven core ASDMs used in practice will be explained. According to Highsmith (2002a:6) these ASDMs are the core group,

as set out in the Agile Software Development Manifesto (Highsmith 2002a:6; Lindstrom & Jeffries, 2004:44). Lindstrom and Jeffries (2004:44) explain that this group is seen as the early initial ASDMs. Furthermore, the core set of ASDMs are relatively settled in practice, and were developed in sequence starting with DSDM in 1994 and ending with the most recent popularised ASDM, LD.

2.4.1 Dynamic System Development Methodology (DSDM)

The DSDM was defined in January 1994 when the sixteen founding members of the DSDM Consortium met for the first time (Hislop *et al.,* 2002:176; DSDM Consortium, 2005). Their goal was to jointly develop and promote an independent Rapid Application Development (RAD) framework, and to expand the proven successes of RAD. A high-level framework was produced that was approved unanimously by the 36 members.

DSDM is not so much a method as it is a framework, and the basic concepts have remained the same although the framework has been refined over time (DSDM Consortium, 2005). According to this source, DSDM "has been found to be applicable in nearly every technical and business environment where systems are needed quickly". Abrahamsson *et al.* (2002:63) state that the main idea behind DSDM is to keep time and resources fixed while adjusting functionality accordingly, and keeping requirements in mind (see Figure 2.1).

The fundamental idea of DSDM differs from the traditional approach where the extent of functionality of a product is fixed, and time and resources must be adjusted to reach this fixed level of functionality. While functionality is allowed to vary, time boxes are used to maintain control. In this way systems can be brought online speedily and without hassle. In this environment, these systems can serve as the basis for further evolution. DSDM is an extension to RAD practices and it at least boasts the best-supported documentation and training of any other ASDM (Highsmith, 2002a:8).

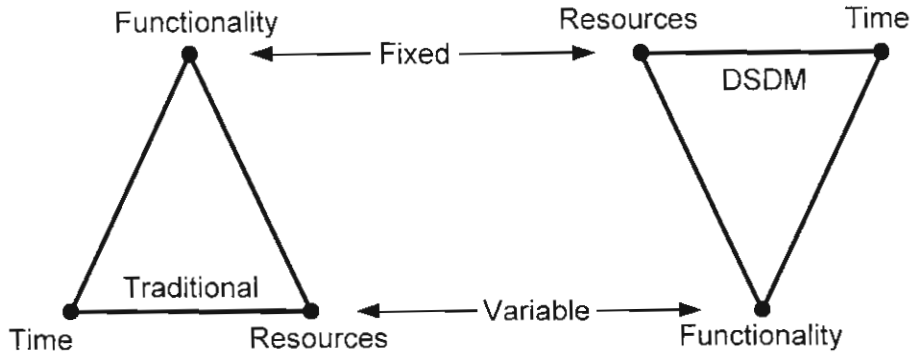


Figure 2.1: Traditional approach vs. DSDM approach (DSDM Consortium, 2005)

The philosophy behind the DSDM framework that drives the thinking process of DSDM developers (DSDM Consortium, 2005) involves the following: Firstly, development can be incremental, which means that the whole development process can be defined in increments (pieces). Each increment can then be designed, developed, tested and deployed. Secondly, development is seen as a team effort where the knowledge of IT professionals and customers are combined. Thirdly, the available resources must initially be spent to develop the most important business requirements. Lastly, to deliver a product of high quality, the organisation must be technically advanced and quick to meet demands.

The DSDM is "lightweight", meaning that it does not focus on documentation. One of the principles of this methodology is the importance of collaboration, i.e. the use of prototypes to capture information rather than numerous documentation (Highsmith, 2002a:8).

The DSDM offers a more complete, defined development process like the most known ASDMs.

The DSDM identifies five distinct phases: feasibility study, business study, functional model iteration, design and build iteration, and implementation.

These are preceded by the pre-project phase and concluded with the post-project phase, as seen in figure 2.2.

Pre-project: This phase ensures that everything is in place and set up correctly, that funding is available, and that the project is ready to begin successfully.

The feasibility and business study: Both these studies are time boxed and done sequentially. The business study usually takes a month where the feasibility study usually takes a few weeks. During the feasibility study the primary goal is to determine, whether the DSDM is the right approach for developing a specific project (Hislop *et al.*,2002:176; Cohen *et al.*, 2003:19).

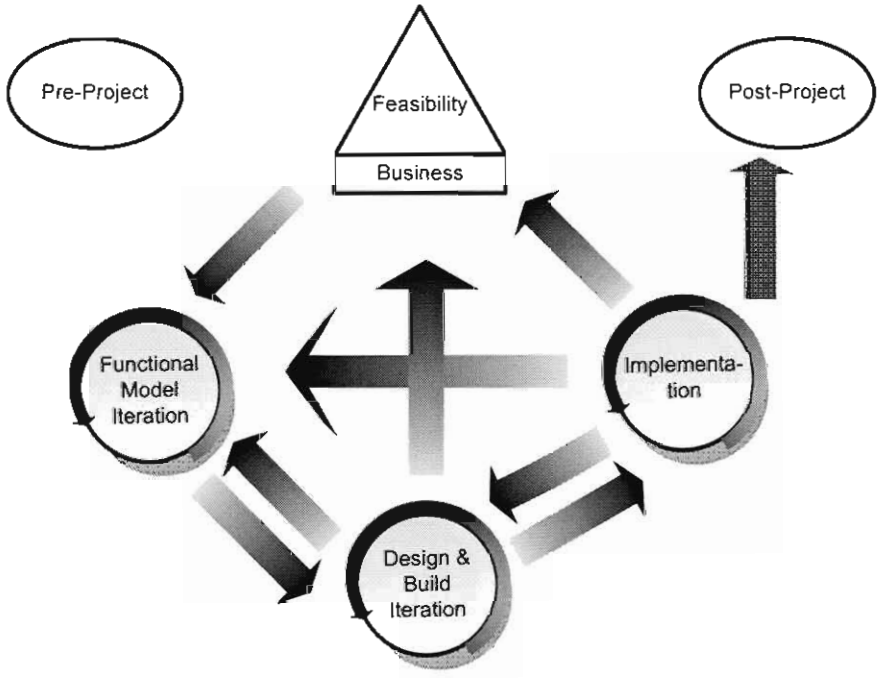


Figure 2.2: The DSDM Lifecycle (adapted from DSDM Consortium, 2005)

In evaluating the type of project, with people and organisational issues as primary concerns, a decision should be made whether the DSDM should be used to develop the project (Abrahamsson *et al.*, 2002:64). According to these authors, the feasibility study is also concerned with the technical and technological possibilities of developing the project, and the risks that may be

involved.

In the business study phase the essential business and technological characteristics are analyzed and prioritized (Abrahamsson *et al.*, 2002:65). This phase consists of working together, using facilitated workshops attended by "empowered and knowledgeable staff who can quickly pool their knowledge and gain consensus as to the priorities of the development" (Cohen *et al.*, 2003:19).

As a result, the business area is defined, describing the affected business processes with their information needs, markets and identified users. Early client identification ensures early customer involvement. Another output in the business study phase is the systems architecture definition, which is the "first system architecture sketch" that has the ability to change as the project develops (Abrahamsson *et al.*, 2002:65). The last output is the outline prototyping plan, which describes the "prototyping strategy for the following stages, and plan for configuration management" (Abrahamsson *et al.*, 2002:65).

If the DSDM is appropriate for the proposed project, the business study scopes the overall activity and sets the framework for both technical and business activities (Hislop *et al.*, 2002:176). After these two phases the high-level requirements are base lined, system architecture is outlined and the functional and information models are produced (Hislop *et al.*, 2002:177).

Functional model integration: The primary concern is to build on the high level of processing and information requirements explained and identified in the business study phase (DSDM Consortium, 2005). Abrahamsson *et al.* (2002:65) explain this phase as the first "iterative and incremental phase". During every iteration, the approach is firstly planned, reviewed and then analysed in order to be applicable in subsequent iterations. The experience gained through coding, analysis and prototype building is used to improve the analysis model. The functional model is produced containing the analysis

models and prototype code.

The functional model provides four outputs (Abrahamsson *et al.*, 2002:65):

- *Prioritized functions:* Is the prioritized list of functions delivered at the end of each iteration.
- *Functional prototyping review documents*: Collecting comments from users about the current increment that can be used for other increments.
- *Non-functional requirements*: Requirements that should be met during the next phase.
- *Risk analysis for further development*: Important document for the functional model iteration phase, because problems will be more difficult to correct from the next phase onwards.

Cohen *et al.* (2003:20) states that this phase as well as the design and build phase have a common process:

- Identify what is to be produced
- Agree on how and when to do it
- Create the product
- Check if it has been produced correctly

Design and build iteration: The prototypes from the functional model iteration are completed, combined and tested to create a system of sufficient internal and external quality to be safely released to the users (Cohen *et al.*, 2003:20). The output of the design and build phase is a tested system that meets at least the most important requirements set by users. The design and build iteration is iteratively. After the users reviewed the design and functional prototypes, all further development is based on the user's comments and requirements (Abrahamsson *et al.*, 2002:66). Just like other agile approaches, testing is not a distinct phase, but very important and woven throughout the DSDM Lifecycle (Hislop *et al.*, 2002:177).

Implementation: In this phase, the system is implemented within the user organisation, and responsibility for operation is transferred to the users (Hislop *et al.*, 2002:177). An increment review document is created during this phase in which the state of the system is discussed. At this stage the system can be either complete, meeting all requirements, or incomplete where some functionalities may be missing, or only some requirements (not even primary requirements) are met. If the system has not been complete, the functional model iteration, design and build iteration, and implementation phases are repeated until the system has been fully completed (Cohen *et al.*, 2003). If the implementation is done over a period of time, this phase may also be iterated (Abrahamsson *et al.*, 2002:66).

The output of the implementation phase is a user manual, explaining how to gain maximum usage of the system, and a project review document, which summarizes the outcome of the project and explains the reason of potential further development based on the project results.

Abrahamsson *et al.* (2002:66) defines four possible courses of development for the DSDM. Firstly, if the system meets all requirements, no further work needs to be done. Secondly, if some requirements were not met because they were only discovered during the development stage, the process may be repeated. Thirdly, if some less-critical function has to be omitted, the process may be repeated from the functional model iteration phase. Lastly, if some technical issues had to be ignored due to time constraints, they may be addressed by iterating again, starting from the design and build iteration phase.

The key is delivering what the business needs when it needs it. This is done by using the various techniques in the framework and flexing requirements. The aim is always to address the current and imminent needs of the business rather than to attack the perceived possibilities (DSDM Consortium, 2005).

Post-project: The main concern is maintenance of the system that has been

implemented. According to the DSDM Consortium (2005), maintenance is done by keeping the project solution operating effectively.

The DSDM has been applied in small and large projects, and also used in combination with other ASDMs (DSDM Consortium, 2005). While the DSDM is continuously evolving within the consortium, no identifiable external research is done by other people except the DSDM authors (Abrahamsson *et al.*, 2002:68).

2.4.2 Scrum

Ken Schwaber and Jeff Sutherland developed Scrum in 1995, but it was firstly described in 1996 (Cohen *et al.*, 2003:13) as a process that accepts the unpredictability of the development process with a "do what it takes mentality", and it has been implemented successfully by numerous independent software vendors.

Scrum, just like XP, is a relatively settled ASDM and a more widely used ASDM in practice. The name "Scrum" is borrowed from the game of rugby. A scrum takes place when eight players of each team, called the forwards, bundle together, and push and shuffle against the opponents for possession of the ball. To get the ball, the one team must displace the other from its current location.

The primary idea of Scrum is that system development involves requirements, resources, technology as well as time constraints, which are likely to change during development. This changing environment makes the development process very complex and unpredictable. The system development process requires flexibility and adaptability to suitably respond to the changes during development (Abrahamsson *et al.*, 2002:27).

According to Abrahamsson *et al.* (2002:27), Scrum is an "empirical approach applying the ideas of industrial process control theory to system development resulting in an approach that reintroduces the ideas of flexibility, adaptability

and productivity". Scrum focuses on producing a system that is flexible by using a team to produce such a system within a constantly changing environment.

Scrum is primarily concerned with a few key management tasks and not so much on how the product is actually constructed (Good, 2003:18). Projects are divided into iterations called "sprints", that take 30 days or less, in which a set of features is delivered. The management of the projects progress takes place in the method called "scrum" also known in some cases as "brain storming", where a daily meeting is held for 15 minutes by the management team (Good, 2003:20).

Abrahamsson *et al.* (2002:28) explains the Scrum process by using three distinct phases, i.e. the pre-game, development and post-game phase as described in Schwaber and Beedle's book *Agile Software Development with Scrum* written in 2002. In this study, the researcher will explain the Scrum lifecycle by using the figure on Control Chaos (2005) as sited on the World Wide Web. This figure is the same as the figure in the book by Schwaber and Beedle (see figure 2.3).

The product backlog (see figure 2.3) contains the body of work required during the entire project. This includes requirements gained from software developers, customers and experts. All requirements are prioritized in a descending order of importance. Due to the ever-changing environment, the product backlog must be constantly updated and prioritized as new requirements are identified. Abrahamsson *et al.* (2002:29) explains the product backlog as part of planning – a sub phase of the pre-game phase. The planning "includes the definition of the system being developed, project team, tools and other resources, risk assessment and controlling issues, training needs and verification management approval". The other sub-phase of the pre-game phase according to Abrahamsson *et al.* (2002:29) is the architecture phase, which includes the changes as well as those problems the changes may cause in implementing the product backlog if the implemented system requires enhancement. A

design review meeting is held to implement these changes.

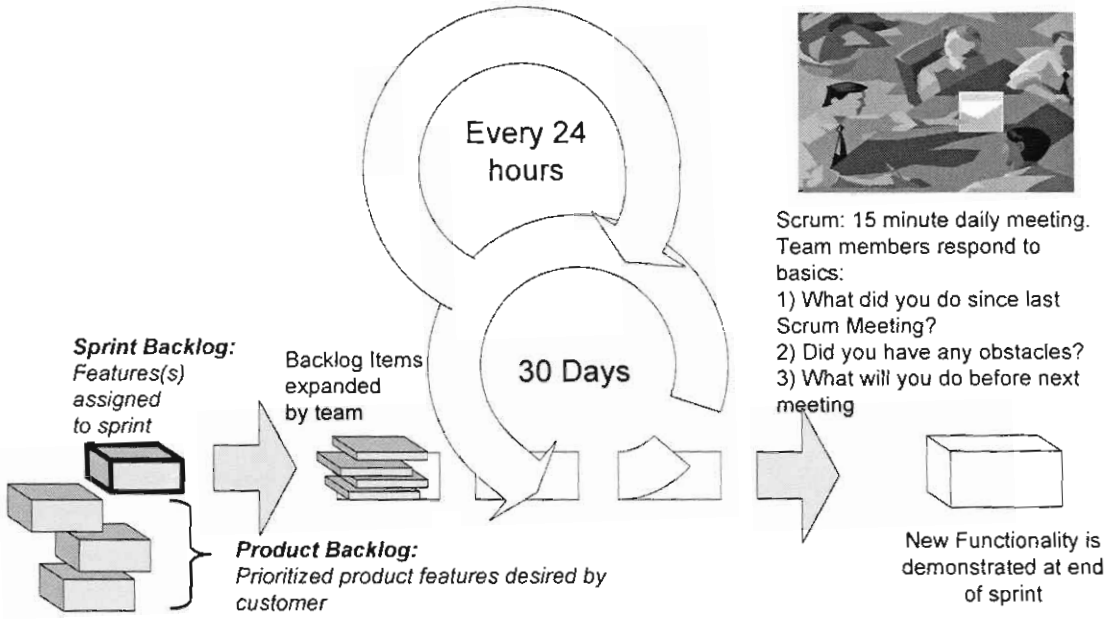


Figure 2.3: The Scrum Lifecycle (Control Chaos, 2005)

A "sprint" is a period of up to 30 days where sectioned tasks will be performed to create deliverables which satisfy the requirements set by users, managers and experts (Huijbers *et al.*, 2004:17). Because development is incremental, each sprint includes traditional phases of software development namely requirements, analysis, design, evaluation and delivery (Abrahamsson *et al.*, 2002:30). There can be as many as eight prints when Scrum is used to develop a system.

Before a sprint is undertaken, the team should do some pre-sprint planning, which includes identifying the tasks necessary to reach the defined sprint goal. These identified requirements and tasks are moved from product backlog to sprint-backlog to be completed during the next sprint (Cohen *et al.*, 2003:14).

The sprint backlog is the starting point for every sprint, which contains all the tasks and requirements that ought to be completed during the current sprint

(Cohen *et al.*, 2003:14). The tasks that should be performed during the current sprint are selected by the Scrum team, the Scrum master and product owner during pre-sprint planning (also known as the sprint planning meeting), using the prioritized list of the product backlog (Abrahamsson *et al.*, 2002:33). The Scrum master is responsible for the project's success. This includes sticking to the rules, values, process and practices of Scrum. According to Abrahamson *et al.* (2002:28) the sprint backlog and sprint are seen as part of the development phase during which environmental and technical variables, which may change during the development process, are observed and controlled. This keeps the team focused on the tasks.

Every morning, a short meeting of approximately 15 minutes is held to keep track of the development process. During each meeting, the Scrum team specifies what has been done since the last meeting, and discuss what should be done before the next meeting takes place (Abrahamsson *et al.*, 2002:34). During these meetings problems are identified and solutions suggested to keep the team focused on the goal. These meetings can also take the form of short and powerful stand-up meetings where definite problems can be discussed and fast solutions found.

After each sprint, a post-sprint meeting or sprint review meeting (Abrahamsson *et al.*, 2002:34) is held to analyze the progress and to demonstrate the system to management, customers and the product owner. This meeting may result in new requirements to improve the system. These new requirements are added to the prioritized product backlog and sprint backlog. The next sprint is then planned, based on using the prioritized product backlog and sprint backlog.

Cohen *et al.* (2003:14) summarize the key principles of Scrum:

- "Small working teams that maximize communication, minimize overhead, and maximize sharing of tacit, informal knowledge
- Adaptability to technical or marketplace (user/customer) changes to ensure

the best possible product is produced

- Frequent builds, or construction of executables, that can be inspected, adjusted, tested, documented, and built on
- Partitioning of work and team assignments into clean, low coupling partitions, or packets
- Constant testing and documentation of a product as it is built
- Ability to declare a product done whenever required"

According to Schwaber and Beedle (2002:59), Scrum can be adopted in existing projects and new projects. Delivering a new project using Scrum, the authors explain that a product backlog must firstly be built by working with the team and customers for several days. The first sprint will then involve key pieces in system development. These key pieces include an initial system framework, technological requirements and business functionality. The sprint will contain the tasks setting up the team roles, building management practices as well as tasks that will fulfil the sprint goal. As the Scrum team members work with the sprint backlog, the product owner works with the customers to build a more comprehensive product backlog. This will enable them to plan the next sprint after the first post-sprint meeting. The post-sprint meeting(s) is seen as part of the post-game phase (Abrahamsson *et al.*, 2002:28).

Scrum can be adopted in an environment where a project with its own technology already exists and in cases where teams are struggling to cope with growing technology and requirements. During the first sprint, user functionality should be demonstrated on the existing system technology (Schwaber & Beedle, 2002:59). During the short meetings, stand-up meetings or scrums held every day, problems are identified or solved. This helps the team to believe in its own abilities, and the customer to believe in the team (Abrahamsson *et al.*, 2002:35). After the first sprint, a post-sprint meeting is held where a decision is made whether the team should continue with the project. If this is agreed upon, a pre-sprint meeting is held to identify tasks to be completed during the next

sprint.

2.4.3 Extreme Programming (XP)

XP was first introduced in 1996 by Kent Beck while serving as project leader on Chrysler Comprehensive Compensation (C3) – a long term project to rewrite Chrysler Corp's payroll application – and was further popularised by Kent's book *Extreme Programming Explained: Embrace Change,* written in 1999 (Copeland, 2001). Numerous articles published subsequently further popularized XP. It is by far the most popular ASDM to emerge in resent years (Highsmith, 2002a:7; Cohen *et al.*, 2003:12; Lindstrom & Jeffries, 2004:43). XP owes most of its popularity to developers' disenchantment with traditional methods that do not work in certain environments. Developers started looking for something new, something extreme, and something that will work in a constantly changing environment.

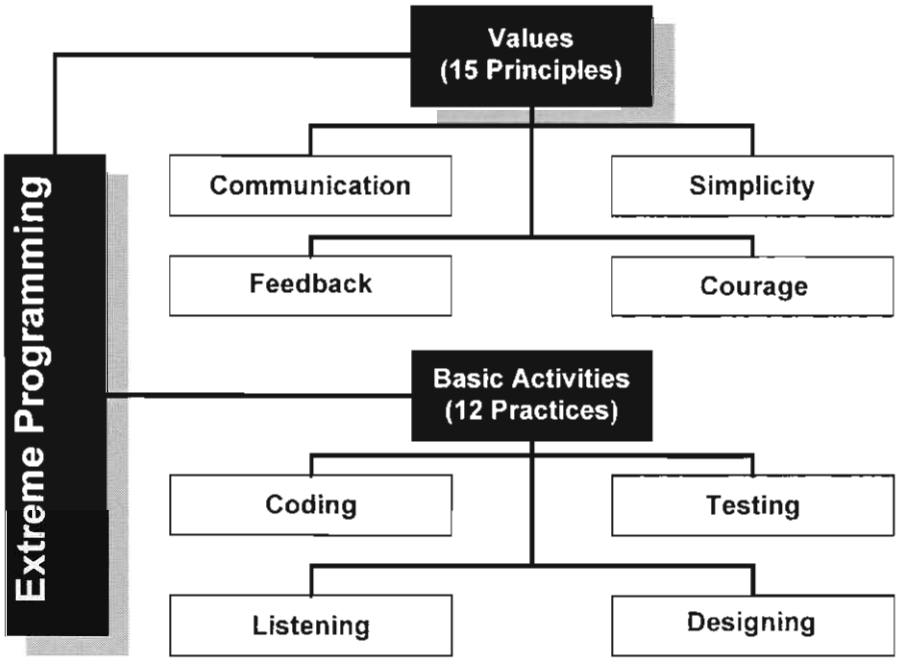


Figure 2.4: The XP Structure (Hislop et al., 2002:173)

XP can best be explained in terms of its structural components, as shown

above in figure 2.4. The four values, *communication*, *simplicity*, *feedback*, and *courage,* yield *fifteen principles* that are unique to XP project development. The four basic activities, *coding*, *testing*, *listing*, and *designing,* can be viewed as XP's "backbone", which forms the basis of the *twelve core practices*.

Four values of XP

According to Lindstrom and Jeffries (2004:50), the values of XP can be used to test if the methodology fits the project, team and organisation. The teams' actions are guided by these values. Lindstrom and Jeffries (2004:45) state that XP is the only ASDM that is "explicit in its values" - that yields principles - and it's practices, as seen in figure 2.4. This explicit combination gives guidance on how to react if the practices do not work (using the values), and what to do (using the practices) (Lindstrom & Jeffries, 2004:45).

Communication: XP is considered "lightweight" because it focuses exclusively on communication between team members, as well as between the team and its customers. The communication should be of high quality for both team members and customers. Development is guided by clients communicating functional requirements as "stories" written on small file cards (Hislop *et al.*, 2002:172), named "story cards".

Simplicity: This requires that system developers and designers build the simplest system that will satisfy the requirements set by sponsors and business users. As the requirements are implemented in the evolving system, system designers and developers must be cautious not to make the design to complex while implementing the necessary modifications. System modification includes the improvement of code structure while preserving system functionality (Hislop *et al.*, 2002:173).

Feedback: Feedback depends on time, because it can take a few minutes or even days. Constant involvement of customers, system designers and

developers causes immediate feedback on the status and progress of the system being developed (Hislop *et al.*, 2002:173). Feedback plays a large role in XP, e.g. where customers define their requirements on story cards, and developers estimate the correct approach to give immediate feedback that will be of value for customers on the work they will do to satisfy these customer requirements. Continuous testing provides programmers with rapid feedback of errors within the design code. Pair programming is also a continuous feedback loop (Hislop *et al.*, 2002:174).

Courage: The developers are responsible for developing a system that is simple, user-friendly, and fulfils most requirements. Creating the simplest design may require courageous decisions, such as throwing out large chunks of code or re-engineering the system to eliminate duplicate code (Hislop *et al.*, 2002:174).

Hislop *et al.* (2002:173) mentions fifteen principles that support XP's values:

- Assuming simplicity
- Incremental change
- Embracing change
- Quality work
- Teaching learning
- Small initial investments
- Playing to win
- Concrete experimentation
- Open, honest communication
- Working with people's instincts
- Accepted responsibility
- Local adaptation
- Travelling light

- Honest measurement
- Rapid Feedback

The four basic activities of XP

Coding: In XP coding is seen as a learning activity. Pair programming is one of the unique practices identified by XP, where two programmers work on the same PC while learning from each other to develop accurate code in less time. "Coding helps the developer to test his/her thinking process, because if the thinking process is correct the code will do what it is designed to do." (Hislop *et al.*, 2002:174).

Testing: Testing is a continuous process throughout system development, as new requirements are implemented into the evolving system. Developers must listen to clients in order to know which requirements should be tested (Hislop *et al.*, 2002:174). Testing every completed task insures that the system design is correct and that all tasks are up to date.

Listening: If system designers and developers do not know how to listen to users, they will not know how to design the system. XP simplifies listening by the use of story cards. It is important for developers to listen to customers' stories so they can help refine these in order to know what should be tested and developed (Hislop *et al.*, 2002:174)

Designing: In XP designing is a continuous activity of incorporating new tasks and requirements into the evolving and already existing system. According to Hislop *et al.* (2002:174), XP relies on a metaphor (brief description that conveys the system's main function attributes) to guide its design.

The twelve core practices of XP

XP practices can be grouped into three different cycles (see figure 2.5). The

outer cycle reflects the practices that affect all the project participants, the middle cycle relates to the work of the development team, and the innermost cycle relates to the work of the developers (Lindstrom & Jeffries, 2004:46).

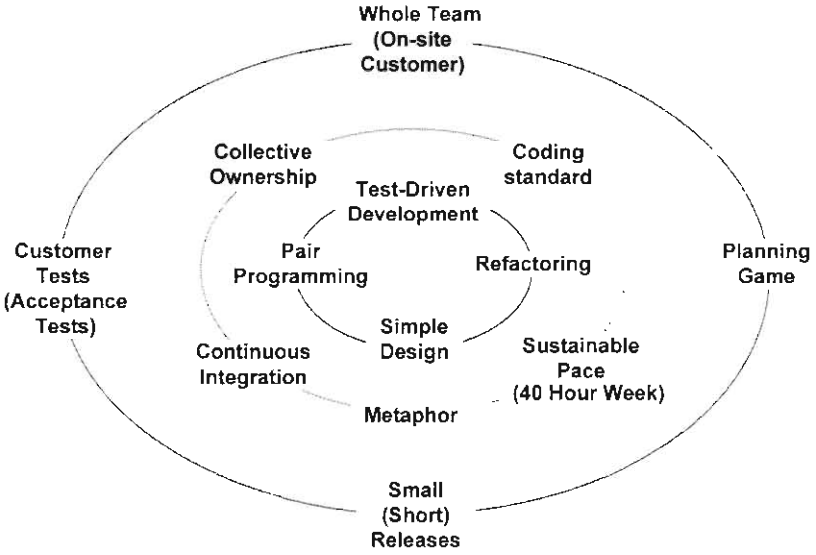


Figure 2.5: Practices and the main cycles of XP (adapted from Jeffries, 2001; Lindstrom & Jeffries, 2004:46)

The XP lifecycle (see figure 2.6) include the twelve core practices, which will be explained. In order to gain a better understanding of the XP lifecycle and how the twelve practices of XP integrate into it, refer to figure 2.6.

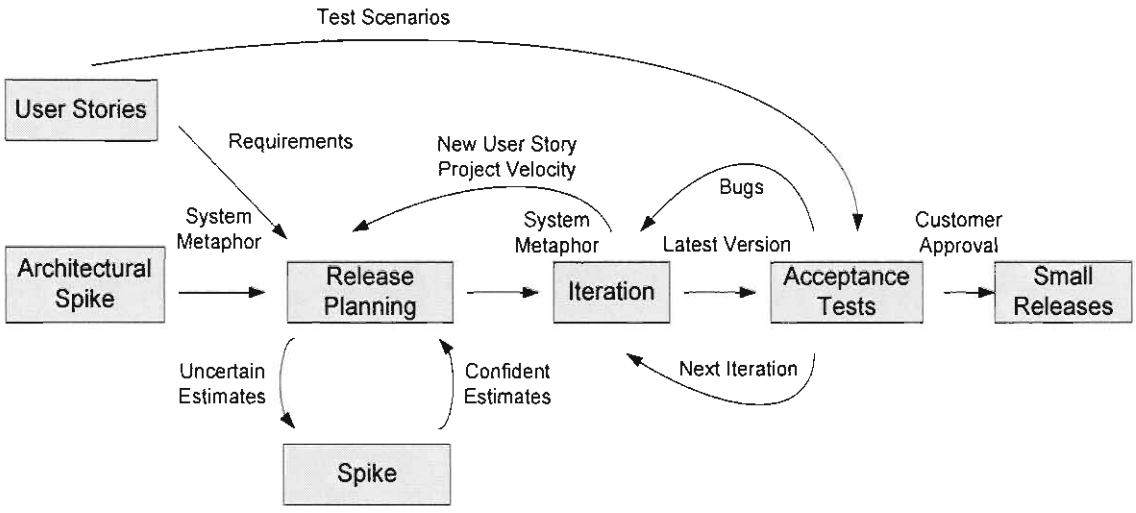


Figure 2.6: The XP Lifecycle (adapted from Wells, 2000)

- *The Planning Game:* As each iteration starts, managers, customers, and developers meet to flesh out, prioritize and predict what should be accomplished before the next release, and estimate the requirements for the next release (Cohen *et al.*, 2003:12; Lindstrom & Jeffries, 2004:47). Lindstrom and Jeffries (2004:47) explain the two key planning steps, namely release planning (the customer presents the desired requirements to system programmers, and the programmers estimate their difficulty) and iteration planning (the team is given direction every few weeks as the system develops). A release is broken into iterations of one to three weeks each. The requirements are called user stories that are captured on story cards in a language all parties will understand (Cohen *et al.*, 2003:12). In this practice the user basically lists the requirements of the system.
- *Small Releases:* According to Lindstrom and Jeffries (2004:47), XP teams practice releases in two ways. Firstly, in every iteration the team produce tested, running software that is of value to customers. Secondly, XP teams release software of business value to their end users as frequently as possible. Development is incremental, which means that a basic system is put into production quickly, where new releases are implemented at least every month until the required (whole) system is up and running (Abrahamsson *et al.*, 2002:23; Good, 2003:23).
- *Metaphor:* Managers, customers and programmers define a metaphor or a set of metaphors which guides all development by describing how the system works (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12; Lindstrom & Jeffries, 2004:49). In short, it guides developers through the development process via requirements, and it explains the system's behaviour.
- *Testing (customer tests and test driven development):* There are two types of tests that are carried out. Firstly, unit tests are done where programmers ensure that the code written does what they expect it will do. In this way, programmers get immediate feedback on their progress (Lindstrom &

Jeffries, 2004:48). Secondly, acceptance tests are done by developers. This involves written acceptance tests for coding the application to ensure that the system does what the user expects it to do (Cohen *et al.*, 2003:12).

- *Simple Design:* Developers are urged to start simple and keep it simple, although new requirements are identified and modifications should be made during development (Lindstrom & Jeffries, 2004:48). The system should be designed and implemented as simple as possible. The reason for using simple design is to create a system that is easy to use, maintain and manage.
- *Refactoring:* When a design is no longer appropriate and of value, it should be changed. Refactoring involves improving the design by keeping the system simple through removing duplication, improving communication and adding flexibility (Abrahamsson *et al.*, 2002:24; Lindstrom & Jeffries, 2004:48).
- *Pair programming:* This involves two programmers programming in front of the same PC (Abrahamsson *et al.*, 2002:24; Lindstrom & Jeffries, 2004:48). This practice ensures effective coding with fewer errors in a shorter time, while the one programmer learns from the other. This practice is also effective if one programmer becomes sick, because the other can continue without time being lost. While one programmer programmes, the other "thinks more strategically" about where the approach will work, and about ways to simplify the design (Avison & Fitzgerald, 2003:444).
- *Collective ownership*: Every developer has ownership of all development documents, and can make modifications anywhere and at any time while system development takes place (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12). The benefits of using collective ownership are increased code quality and reduced defects, as well as elimination of code duplication by different programmers (Lindstrom & Jeffries, 2004:49).
- *Continuous integration:* Developer integrate a new piece of coding as soon and as often as possible (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12). Every time a task is completed, the completed task is integrated

into the system after which tests are ran, that must be past, in order for the new changes in the code to be accepted (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12; Lindstrom & Jeffries, 2004:48).

- *Forty hour week:* Developing software is very demanding. Therefore, as a rule XP states that team members work no more than 40 hours per week. Overtime is allowed, but not for more than two subsequent weeks (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12). This prevents programmers and developers burning out, as well as negligent work that may harm the project.
- *On-site Customer:* Using XP, the customer is on-site at all times, contributing to development, answering questions, performing acceptance tests, and ensuring that the development progresses as expected (Abrahamsson *et al.*, 2002:24; Good, 2003:23; Cohen *et al.*, 2003:12). The on-site customer ensures that the developers stay focused on the requirements. If they do lose focus, the customer is there to get the developers back on track to satisfy requirements.
- *Coding standards (open workspace):* The programmers of an XP project follow a common coding standard so that all code that emphasize communication, looks as if it is written by one individual (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12).

The rhythm of an XP project

According to Lindstrom and Jeffries (2004:45), an XP project has a rhythm that proceeds in iterations of two weeks. During each iteration a set of requirements is developed and tested. Figure 2.7 shows the activities of the programming team and customers during the iterations of an XP project. As the project steadily progresses, the customer chooses when the entire project (with maximum functionality) is delivered (Lindstrom & Jeffries, 2004:45).

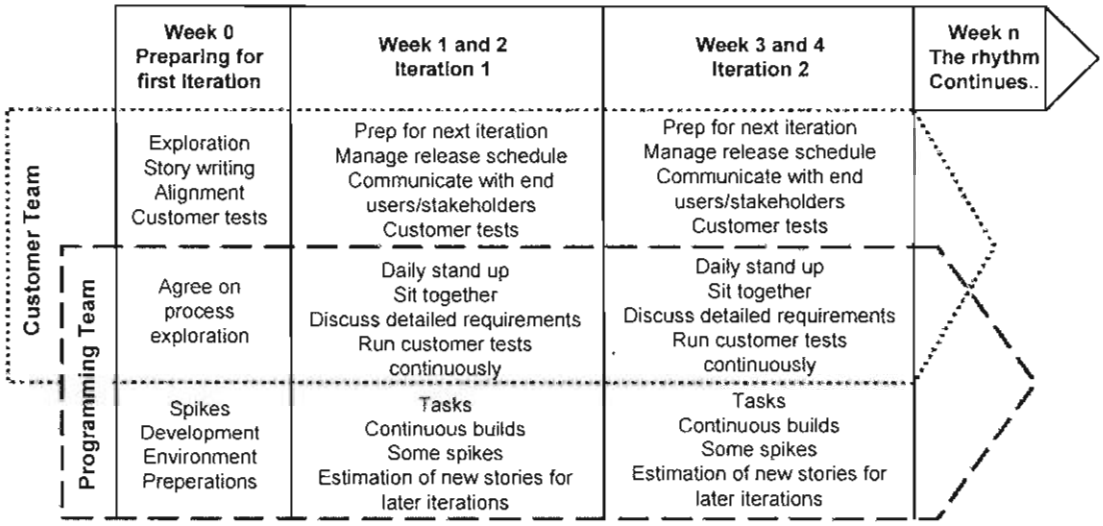


Figure 2.7: Rhythm of an XP project (Lindstrom & Jeffries, 2004:46)

Lindstrom and Jeffries (2004:50) state that the most commonly debated question regarding XP is whether it can be applied successfully to a particular type (different environments) of project. Experience proves that system development is effected and limited by the "characteristics of the project, the people on the team, and the organisation in which they work" (Lindstrom & Jeffries, 2004:50). To evaluate whether the XP practices can help a team achieve greater success, consideration must be given to these limitations.

2.4.4 Feature Driven Development (FDD)

Feature Driven Development (FDD) was created by Jeff De Luca and Peter Coad in 1997 and later popularized in their book written in 1999, *Java Modelling in Colour with UML*. This ASDM was created when De Luca and Coad were brought in as consultants on a project in trouble (the large lending system project at United Overseas Bank in Singapore). Coad applied feature-oriented development techniques on the project while De Luca used a streamlined, lightweight process framework (Hislop *et al.*, 2002:175). Coad and De Luca merged their concepts into an ASDM that became known as FDD to save their highly complicated Singapore project from failing (Hislop *et al.*,

2002:175; Cohen *et al.*, 2003:17). The previous developers spent two years on the same project, writing over 3 500 pages of documentation without any code (Highsmith, 2002a:5) and declared the project undoable (Hislop *et al.*, 2002:175; Cohen *et al.*, 2003:17). FDD was applied to the failing project and began delivering a product in increments to a surprised customer within two months (Hislop *et al.*, 2002:175; Highsmith, 2002a:5).

FDD relies on a basic architecture that is represented as UML class diagrams, which are developed early in the project. The FDD ASDM primarily focuses on the design and building phases of the software development process and it does not need a specific process model to be successful (Abrahamsson *et al.*, 2002:47). According to Palmer and Felsing (2002:35), FDD is built around a core set of "best practices" that compliment and reinforce each other.

Best practices in FDD include (Palmer & Felsing, 2002:35):

- *Domain object modelling*: The problem domain is explained and explored to deliver a framework where features can be added.
- *Developing by feature*: The progress is tracked via a list of small functionally decomposed and client-valued functions.
- *Individual class (code) ownership*: The responsibility of performance, consistency and conceptual integrity of a class is assigned to an individual.
- *Feature team* are small, dynamic teams.
- *Inspection* is done by using the best-known defect detection mechanisms.
- *Regular builds* ensure that there is always a running basic system available to which new features can be added.
- *Configuration management*: The latest versions of each source code file are identified and tracked through configuration management.
- *Progress reporting* involves reporting on all completed sections at organisational level.

The lead designers decompose these business practices into feature to plan,

design and code these features (Abrahamsson *et al.*, 2002:47). Features are small items useful for users, just like story cards used in XP, written in an understandable language for all parties that should not take longer than two days (Cohen *et al.*, 2003:18). FDD encapsulates best practices and incremental development to manage and monitor the development process and, when completed, deliver features to customers in two week cycles (Abrahamsson *et al.*, 2002:47; Hislop *et al.*, 2002:175).

Cohen *et al.* (quoted in Highsmith, 2002c:273) explain some core values that would work best for developing a project using FDD. These values are:

- A system for building systems is necessary in order to scale to larger projects.
- A simple, well-defined process works best.
- Process steps should be logical, and their value immediately obvious to each team member.
- "Process pride" (developers believing that their process will work, although a process exist that will work much better in the current situation) can keep the real work from happening.
- Good processes move to the background so team members can focus on results.
- Short, iterative, feature-driven lifecycles are best.

The FDD ASDM consists of five sequential processes, including techniques, guidelines, rolls, goals, artefacts, timelines and methods that can be used in FDD project development. The five processes are illustrated in figure 2.8:

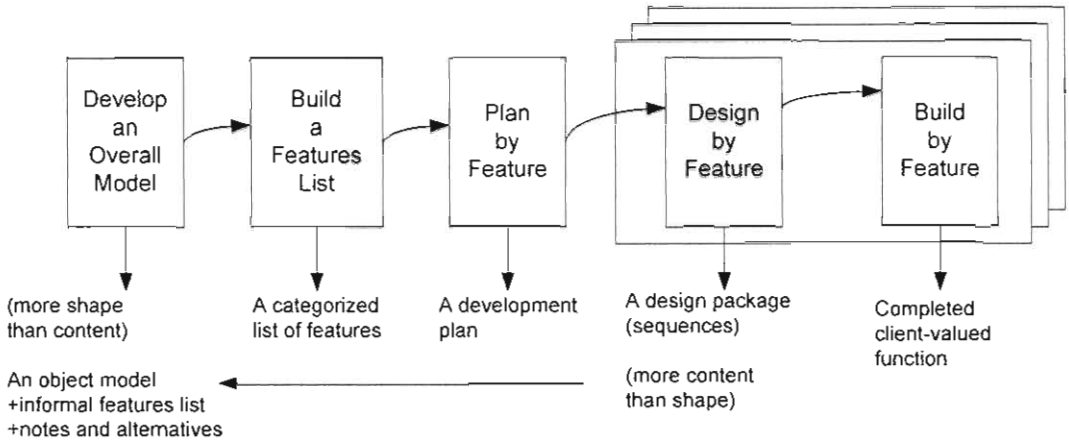


Figure 2.8: FDD process (adapted from De Luca, 2005:10)

Develop an overall model: In figure 2.8, the FDD process begins with developing a model where any user, client, sponsor, business analyst, or a combination of these role-players already know what the requirements, scope and goals for developing the system will be. These domain experts present a "walkthrough" version of the project in which the chief architect, who is responsible for the overall system design, and members of the team are informed of the primary requirements and system description (Abrahamsson *et al.*, 2002:48, 52).

The overall domain is divided into different domain areas. A detailed "walkthrough" is held where every domain is concerned to produce object models for each domain area (Abrahamsson *et al.*, 2002:48). An overall model is constructed by choosing the appropriate object models for each domain.

Build a feature list: During the next step the team identifies features that represent the system. A feature list can easily be built based on the "walkthroughs", identified requirements and object models identified by the development team.

According to Abrahamsson *et al.* (2002:48), the development team list client valued functions in the feature list. These functions (or function group) consist

of major feature sets that represent each of the domain areas. The major feature sets are further divided into feature sub-sets, that "represent different activities within specific domain areas" (Abrahamsson *et al.,* 2002:49) (see figure 2.9). Features longer than ten days are divided into sub-features (Cohen *et al.,* 2003:18). The sponsors and users review the feature list, after division, to determine whether the list is complete and valid.

Plan by feature: The next step is to prioritize the collected feature list into design packages and assign them to chief programmers (Cohen *et al.,* 2003:18). Chief programmers are experienced developers who lead small teams in the analysis design and development of new features. Chief programmers also assign class ownership and responsibility to other individual developers.

Design by feature and build by feature: This iterative process begins when the chief programmer selects a small group of features from the feature set(s) (see figure 2.9). The selected features are planned in more detail; built, tested and integrated iteratively within two weeks (Abrahamsson *et al.,* 2002:49; Cohen *et al.,* 2003:18).

After the successful completion of each iteration, which includes tasks of coding, testing, and integrating, the current iteration becomes part of the main system and the next iteration is started. During this iteration the chief programmer chooses new features from the feature set(s).

To better understand features, feature sets and a feature list as a whole, examine figure 2.9:

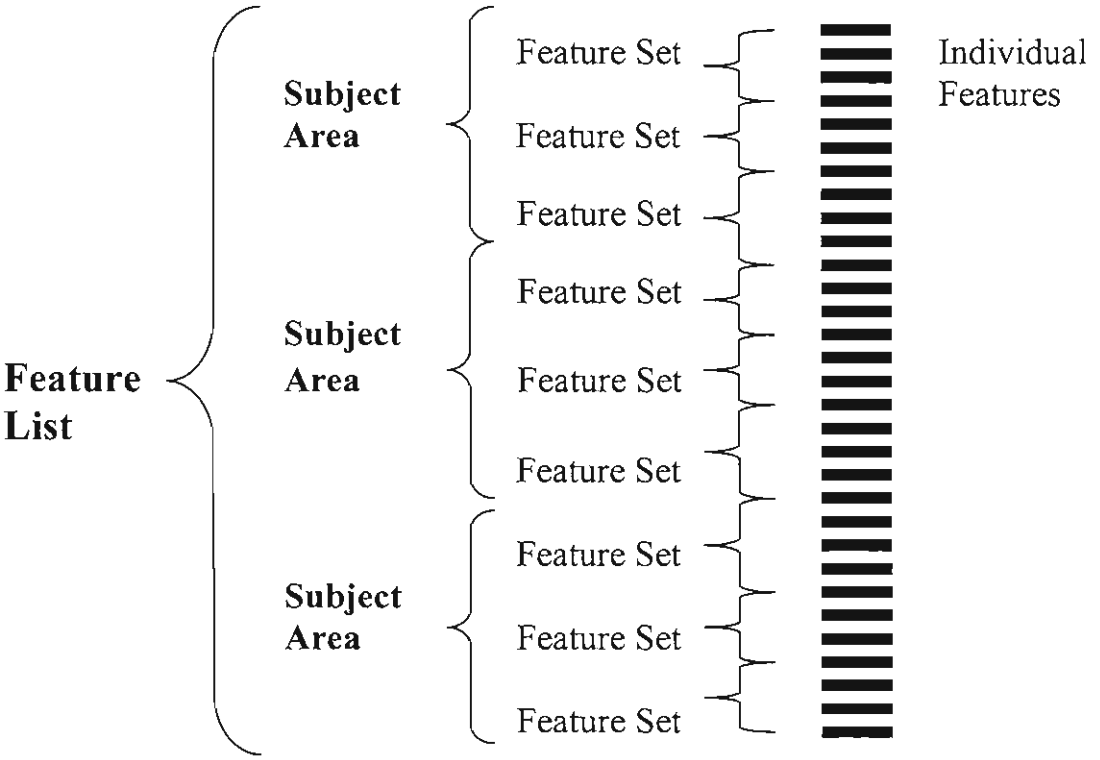


Figure 2.9: Component assembly in FDD (Anderson, 2004:5)

Figure 2.9 explains the component assembly of FDD very simply, where a feature list consist of several subject areas. A subject area consists of several feature sets and a feature set consists of individual features.

The FDD ASDM promises early and frequent delivery of working code in increments with almost no documentation to satisfy customers. Unlike XP, FDD needs an architectural design in the form of class diagrams and analysis of features using sequence diagrams (Hislop *et al.*, 2002:176). Similar to other ASDMs, this ASDM promises early and continuous customer involvement throughout the project (Hislop *et al.*, 2002:176).

FDD has a high success rate with large projects if diverse talent exist within the project, i.e. competent domain experts, developers and chief programmers (Highsmith, 2002a:6). According to Abrahamsson *et al.* (2002:54), FDD is also suitable for new projects, projects in need of code upgrading, and projects that

require the development of a second version. It will be wise to adapt this ASDM in small increments as development progresses (Abrahamsson *et al.*, 2002:54).

2.4.5 Crystal ASDMs

Crystal ASDMs (Crystal Clear) were developed by Alistair Cockburn in 1999, because he believed that one of the major obstacles in product development is poor communication. He aimed to address different kinds of project requirements with different kinds of Crystal ASDMs. Highsmith and Cockburn (2001:120) explains this philosophy as follows: if you replace written documentation with face-to-face interaction, you could improve the likelihood of delivering a system in frequent running pieces and reduce its reliance on documentation. Crystal ASDMs focus on people, the interaction between people and the community, talents and skills, and the process, but primarily on communication (face-to-face interaction).

Crystal project development is incremental with a maximum increment length of four months, but preferably between one and three months (Abrahamsson *et al.*, 2002:37). Crystal ASDMs consist of a family of ASDMs, which gives developers the choice of choosing the most appropriate methodology for a specific project.

In figure 2.10, the Y-axis represents the criticality of the system or the project that must be completed, while the X-axis represents the number of people involved in a project team. In a large team (501 -1000 people), life's criticality of the system is prioritized as the most difficult according to figure 2.10. By adding people to the project, you move to the right on the framework to a darker version of a Crystal ASDM. As the project's criticality increases, the methodology becomes more difficult, and you move upwards on the Y-axis. According to Cohen *et al.* (2003:16), Crystal ASDMs can also be adapted to other priorities such as productivity or legal liability.

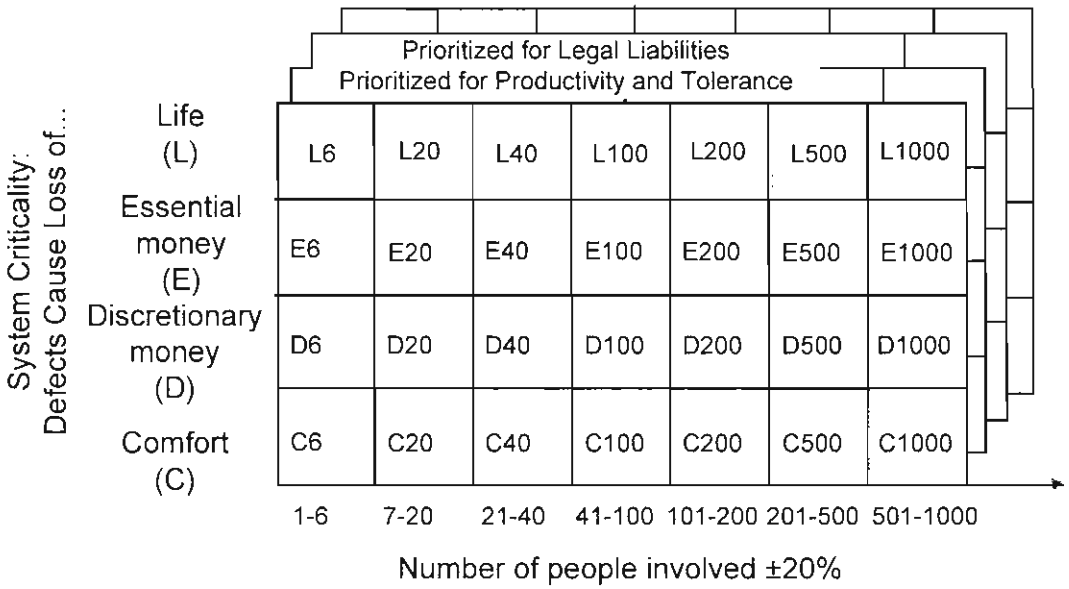


Figure 2.10: Crystal ASDMs framework (Cockburn, 2001; Highsmith, 2002b)

In figure 2.10 the symbols C, D, E and L are an indication of potential loss caused by a system failure (Abrahamsson *et al.*, 2002:36). The critical level symbols stand for: Comfort (C), Discretionary (D), Essential Money (E), and Life (L). Abrahamsson *et al.* (2002:36) explain that the criticality level C indicates a system crash because of defects that cause a loss in comfort for users, whereas L indicates a defect in the critical system that may literally mean a loss of life. Every block within the graphic (X- and Y-axis) of figure 2.10 represents a project category symbol. Abrahamsson *et al.* (2002:37) takes D6 as an example and explain it as a project with a maximum of six people delivering a system of maximum criticality of Discretionary Money (D).

Each Crystal ASDM has a colour that describes its difficulty (heaviness); the darker the colour, the more difficult the ASDM. There are seven main colours; Clear, Yellow, Orange, Red, Maroon, Blue, and Violet. According to Abrahamsson *et al.* (2002:36) Crystal ASDMs suggest choosing the most applicable colour for a project based on its criticality and size

The most commonly used Crystal ASDM is Crystal Clear, followed by Crystal

Yellow, Crystal Orange, Crystal Red, etc. (Cohen *et al.*, 2003:16). The Crystal ASDM used, depends on the degree of importance of communication and the number of people involved.

According to Cohen *et al.* (2003:16) this known set of Crystal ASDMs expands as the ASDM becomes more difficult (hardens) or the project team grows. In this literature study, the researcher will only explain Crystal Clear, because the team that will develop the data warehouse in Chapter 5, will not contain more than six members. Another reason for explaining CC is that, according to Abrahamsson *et al.* (2002:38), CC and Crystal Orange are the two Crystal ASDMs that have been experimented with in practice.

Crystal Clear (CC)

CC is designed for a very small project (category D6 projects) with a maximum team member count of six people, sharing office space because of the importance of communication. However, Abrahamsson *et al.* (2002:38) explain that if communication (face-to-face) and testing is extended, CC can also be applied to E8/D10 projects. According to Cockburn (2005:307), "CC is a highly optimized way to use a small, collaborate team prioritizing for safety in delivering a satisfactory outcome, efficiency in development, and habitability of the working conventions."

CC is part of the family of Crystal ASDMs where every ASDM is identified and characterized by a colour as stated earlier. The more people the team consist of, the darker the colour gets and the harder (heavier) the project becomes. Seven properties should be accomplished in every project. The first three properties are mandatory, while the last four will allow the project to succeed. However, all seven are desired in a project.

Seven properties of Crystal Clear:

1. *Frequent delivery*: CC is an ASDM (explained in par. 2.3), i.e. development is iterative; it is tested every few months (periods should be no longer than four months), and working code must be delivered and implemented into the system. This causes continuous customer involvement, resulting in feedback on the implemented requirements, as well as satisfied sponsors and developers.
2. *Reflective improvement*: Users identify flaws in the system while iterative development and implementation takes place. Requirements that have not been met, are identified and the project team is given time to improve on deficiencies.
3. *Osmotic communication*: CC focuses on face-to-face communication. It would be wise to keep the team working closely together, if possible in the same room, so that all questions and problems can be answered and solved. This will cause a neutral work environment where team members can acquire relative information, just like osmosis.
4. *Personal safety*: Team members and users should have the confidence to offer constructive criticism on the work of other team members, and take responsibility for their own mistakes. This will lead to trust among team members because they are honest with one another.
5. *Focus*: There should not be any distractions that can cause team members to lose concentration, such as long meetings and other activities that require multitasking. This will cause team members to be more focused on their primary objectives and work will be completed much quicker. Focus should be maintained for "two hours a day and for two consecutive days every week" (Huijbers *et al.*, 2004:20).
6. *Easy access to expert users*: Questions associated with quality and design could be answered by available experts who assist the team during system development.
7. *Technical environment with automated tests, configuration management and frequent integration*: It is critical to have a proper technical environment

where testing and controlling tasks, e.g. making backups and merging changes, do not have to be done manually in order to make life easier for developers. This will cause the project to be completed in less time.

Policy standards can be derived from the seven properties of CC. The six policy standards are *(Abrahamsson et al., 2002:39):*

- Incremental delivery on a regular basis (one-three months)
- Progress teaching by milestones based on software deliveries and major decisions rather than written documentation
- Direct user involvement
- Automated regression testing of functionality
- Two user viewings per release
- Workshop for product and methodology timing at the beginning and halfway through each increment

CC has a restricted communication structure and is only suitable for a single team working in one office space. According to Abrahamsson *et al.* (2002:46), CC lacks system validation elements that cause it to not be applicable to life-critical projects. Huijbers *et al.* (2004:21) state that CC values "properties over techniques". This causes team members to use their own techniques to satisfy the seven CC properties. Therefore, there is not a specific list of techniques that need to be used in order to ensure CC's success.

2.4.6 Adaptive Software Development (ASD)

ASD was developed by Jim Highsmith and first documented in his book, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, written in 2000. ASD focuses on iterative software development, with constant prototyping to develop complex, large systems (Abrahamsson *et al.*, 2002:68). ASD also focuses on results, not tasks, which are identified as application components. According to Highsmith (2000:23),

ASD is designed for extreme projects where high speed, frequent change and a high level of uncertainty is the order of the day. This ASDM emphasises change, and change is positive because it prepares customers and developers for the future. There are many projects that are not extreme, but for those who are complicated and extreme, ASD works better than the original and traditional software development approaches (Highsmith, 2000:23).

The traditional project management *Plan-Design-Build* lifecycle is replaced with the *Speculate-Collaborate-Learn* lifecycle in ASD (see figure 2.11). In the traditional approach, uncertainty in the planning phase is seen as a weakness that can evolve in failure. Highsmith (2000:23) states: "The new ASD lifecycle is dedicated to continuous learning that is geared to constant change, re-evaluation, peering into an uncertain future, and intense collaboration among customers, developers and testers." The phases in the ASD lifecycle are named in a way to emphasise the "role of change" in the development process (Highsmith, 2000:23).

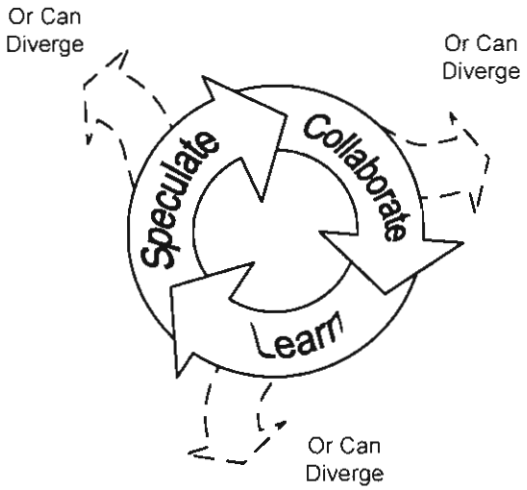


Figure 2.11: The Adaptive Lifecycle (Highsmith, 2000:23)

Highsmith (2000:24) explains *speculation* as the recognition of the uncertain nature of complex problems and encourages experimentation and exploration. It gives developers and designers room to explore, and lets them realize that they are unsure, which will allow them to change their plans without fear.

Speculation means to keep delivery cycles short and encourage iteration, it merely acknowledges the reality of uncertainty and it does not mean that planning is abandoned. Deviations must not be seen as mistakes, but as a learning opportunity. "Speculating allows us to admit that we don't know everything – and once we admit to ourselves that we are fallible, then learning becomes more likely." (Highsmith, 2000:24).

Living in an environment where technology is an ever-changing process, a group of developers cannot possibly know everything. Developers work together using *collaboration* skills, whereas designers make decisions or produce results (Highsmith, 2000:24), to solve complex problems much easier. In designing large and complex problems, a large volume of information must be collected (too large for one developer), analysed and applied to solve the problem.

In order to become adaptive, the organisation and developers must focus on *learning*. They will then have the ability to respond to change that may occur in almost every project. "Learning about oneself – whether personally, at a project team level or at an organisation level – can be painful" (Highsmith, 2000:24). A post mortem is an example document that could be used to determine successes and failures, but it could also become a reason to blame someone instead of a learning tool.

A problem is not a big issue. Developers should learn from their mistakes and problems. Learning is an ongoing process, and mistakes are inevitable.

Highsmith (2000:27) identifies four categories of lessons to be learned by the end of each development cycle:

- Quality of results from the customer's perspective
- Quality of results from a technical perspective
- The functioning of the delivery team and the practices they utilise

- The project status

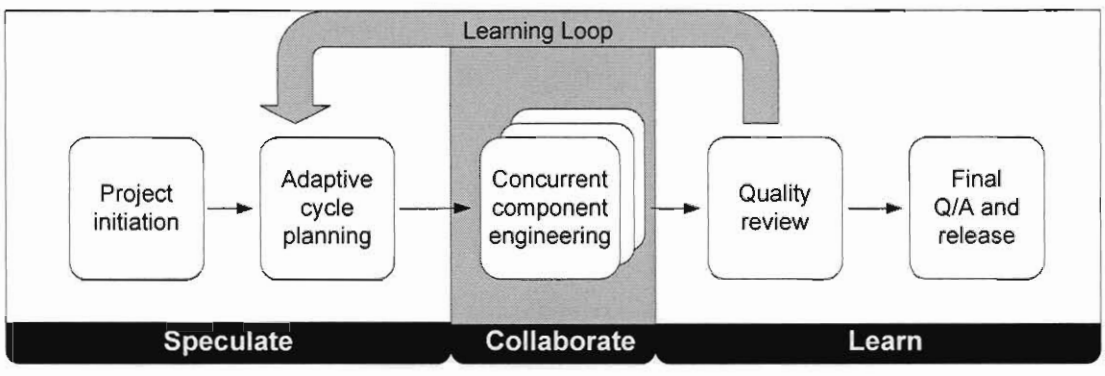


Figure 2.12: Adaptive Lifecycle Activities (Highsmith, 2000:26)

The ASD lifecycle is explained in more detail in figure 2.12.

Speculate

There are seven steps of adaptive cycle speculation (Highsmith, 2000:24):

1. Conduct the project initiation phase
2. Determine the project timebox
3. Determine the optimal number of cycles and the timebox for each
4. Write an objective statement for each cycle
5. Assign primary components to cycles
6. Assign technology and support components to cycles
7. Develop a project task list

Most of the data should be gathered in Joint Application Development (JAD) sessions. A JAD session is a workshop where developers and customers meet to "brainstorm" ideas, discuss product deliverables (features), and to enhance communication (Abrahamsson *et al.*, 2002:71). Initials for small projects can take a week, while large projects may require a "Cycle 0", which involves delivering preparatory deliverables to the customer, but no sections of the application (Highsmith, 2000:26).

According to Highsmith (2000:26), project initiation (*step1*) involves:

- Setting up the project mission and objectives
- Understanding and documenting constraints
- Establishing the project organisation and key players
- Identifying and outlining requirements
- Making initial size and scope estimates
- Identifying key project risks

The timebox (*step 2*) should be based on the resources from project initiation, scope, requirements set by users, feature set(s), and time and budget estimates.

The individual cycle length (*step 3*) depends on two factors: the overall project schedule and the degree of uncertainty. For a small to medium sized project, a cycle length from four to eight weeks is required.

"Each cycle should have its own theme." (Highsmith, 2000:26). Every cycle (*step 4*) has its own milestones to fulfil. It is important to force product visibility that will show all the mistakes, problems and defects in the project. A cycle delivers a workable presentation set of components to the customer, and it makes the product visible to the development team (Highsmith 2000:26). It is also important to remember that testing takes place continuously.

The main concern of component assignment is that every cycle should deliver a visible, acceptable result. Factors that should be taken in consideration when assigning components to cycles (*step 5* and *6*) are (Highsmith, 2000:26):

- Making sure each cycle delivers something useful to customers
- Identifying and managing high-risk items early in the project
- Scheduling components to accommodate natural dependencies
- Balancing resource utilisation

Developers who feel uncomfortable without a task list, could make each component the task target (Highsmith, 2000:27). Additional tasks with no relation to components can also be added. The primary plan of component assignment is a "component breakdown structure" and not a "work breakdown structure" (Highsmith, 2000:27).

Collaborate

Managers are more worried about dealing with collaboration and concurrency than about the details of designing, coding and testing. "Concurrent component engineering delivers the working components." (Highsmith, 2000:27). Concurrency in a project is a critical issue. In large projects, concurrency could be managed by using an advanced adaptive lifecycle, while in small projects it could be managed informally, because team members work closely together. Collaborative development in small teams could be enhanced by using some XP practices like collective ownership and pair programming.

Learn

In order to deliver a project of quality, review practices should be fulfilled. The following review practices should be fulfilled in the learning phase (Highsmith, 2000:27-28):

- Providing visibility and feedback from the customers: this can be achieved by using customer focus groups, which are designed to review the application, explore a working model of the application and record client requirements.
- Reviewing technical quality: Focuses on the technical quality assessment of the product.
- Monitor the team's performance: This can be called the people-and-process review where post-mortems are needed. There are four basic post-mortem questions (Highsmith, 2000:28):
    - What is working?

- What is not working?
- What do we need to do more of?
- What do we need to do less of?

Post-mortems force developers to learn about themselves and explains an organisation's ability to learn.

- Review of project status: The basic questions asked for reviewing the status of a project (Highsmith, 2000:28):
  - What is the status of the project?
  - What is the status compared to our plans?
  - What should that status be?

  At the beginning of each cycle, this review replans the project effort. In a component-based approach, the project status reflects multiple components at different stages of completion.

According to Abrahamsson *et al.*, (2002:70), the learning loop (as seen in figure 2.12), which is gained from repeated quality reviews, forms the basis of further cycles. The quality reviews (quality practices) demonstrate the functionality of software being developed during each cycle. Quality reviews are performed at the end of each cycle and it is important to keep customers involved using JAD sessions.

Characteristics of an adaptive lifecycle (Highsmith, 2000:25)

- *Mission focused:* A mission provides boundaries rather than a fixed destination. Without a good mission statement and refinement process, iterative lifecycles will become a lifecycle with no progress (oscillating lifecycle). Mission statements act as guides that encourage exploration, while mission artefacts provide direction and critical decisions (Highsmith, 2000:25).
- *Component based:* Defines a group of features that is developed during an iterative cycle.
- *Iterative:* "Iterative cycles emphasize 're-doing' as much as 'doing'"

(Highsmith, 2000:25)

- *Timeboxed:* Timeboxing means setting fixed delivery times for iterative cycles and projects. Timeboxing forces a project team and customers to overlook and constantly re-evaluate the project's mission profile (consisting of scope, schedule, resources, and defects). It is about focusing on hard trade-off decisions (Highsmith, 2000:25).
- *Risk driven:* The adaptive lifecycle plans are driven by analysing the risks critical to the project (Highsmith, 2000:25).
- *Change tolerant:* It is the ability to incorporate change as a competitive advantage.

As stated earlier, technology is ever changing and change will be a definite fact during system development. Using adaptive approaches like ASD can make large projects that need extreme development a success by managing change and delivering an up to date product.

2.4.7 Lean Development (LD)

Bob Carette's Lean Development (LD) is derived from the principles of lean manufacturing used during the restructuring of the Japanese automobile manufacturing industry, such as Honda and Toyota, in the 1980's to compete with American motor vehicle manufacturers. LD was later popularised by Mary Poppendieck's book, *Lean Software Development: An Agile Toolkit*, written in 2003.

Lean Manufacturing still used in the automobile industry is concurrent rather than sequential. Decisions are made as late as possible with as much information as possible. Critical decisions are made by the developers themselves. One leader states what the automobile should look like, and constantly explains that to the engineers and developers (Lindberg, 2003).

In Poppendieck's book, she provides 22 tools for converting lean principles into

agile system development practices. In this literature study, only the seven lean principles will be discussed and the 22 tools will only be mentioned.

Just like other ASDMs, LD focuses on customers (people), iterative development, value flow and of accelerating application development speed, but not at the expense of higher defect or cost rates (Highsmith, 2002a:6). According to Cohen *et al.* (2003:18), other ASDMs try to change the development process. To be truly agile, LD should change the way organisations work from the top down. The authors also state that LD is a management philosophy rather than a development process.

Using LD as an ASDM, the key for any organisation is to be more agile than their competitors. This means an organisation that is change tolerant and focuses on risk entrepreneurship. The latter enables companies to turn risk into opportunity and to use the opportunity to their own benefit. A change tolerant organisation causes changes that keep competitors off balance and out of the way (Highsmith, 2002a:6). Every business should build a high tolerant organisation in order to deal with changes. According to Highsmith (2002a:6) there are mainly three goals in LD:

Complete the project in:

- One-third of the time
- One-third of the budget
- One-third of the defect rate

To be lean, the organisation has to think lean. According to Poppendieck (2003:2), "lean thinking" is to "let customers delay their decisions about exactly what they want as long as possible, and when they ask for something, give it to them so fast they don't have time to make up their minds".

The seven principles of LD are guideposts to convert them into agile system development practices.

The seven LD principles include (Poppendieck, 2003:3-7):

1. *Eliminate waste*

   Anything that is not of value to customers are seen as waste. If waste is identified, a campaign should be launched to eliminate it immediately. The seven wastes in software development are (Poppendieck, 2003:3):

   - "Partially done work (the 'inventory' of a development process)
   - Extra processes (easy to find in documentation-centric development)
   - Extra features (develop only what customers want right now)
   - Task switching (everyone should do one thing at a time)
   - Waiting (for instructions and information)
   - Handoffs (tons of tacit knowledge gets lost)
   - Defects (at least defects that are not quickly caught by a test)"

   LD focuses on the elimination of waste by looking at the flow of value from the request to implementation. In order to let value flow, teams should be formed to take each requirement from start to finish as fast as possible.

   Tools: Seeing Waste, Value Stream Mapping (Steindl, 2004; Norton, 2005).

2. *Amplify learning*

   "Great designers understand that designs emerge as they develop a growing understanding of the problem." (Poppendieck, 2003:2). Development is a learning process. Developers discuss what should be done; ways to make it work and they try it. If it does not work, they learn from their mistakes and try again. According to Poppendieck (2003:4), a development is no place for slogans such as:

   - "Plan the work and work the plan
   - Do it right the first time
   - Eliminate variability"

The idea is to adapt to variation by constant feedback to customers and not to eliminate variety as a whole. Because of iterative development, developers can measure the difference between what the customer wants and what the software can do in order to make the correct adjustments. LD focuses on feedback, using short (a week to a month) and full cycle (tested, integrated and deployed code) iterations. According to Poppendieck (2003:4), "Iterative (evolutionary) development is the best approach for software development."

Tools: Feedback, iterations, synchronization, set-based development (Steindl, 2004; Norton, 2005).

3. *Delay commitment*

To delay commitment means to keep all options open as long as system development allows it. The fundamental concept of LD is to "delay irreversible decisions until they can be made based on known events rather than forecasts" (Poppendieck, 2003:4). Having options, make customers delay decisions until they have enough accurate information to make a right (not predicted) decision.

Ways to keep options open in system development (Poppendieck, 2003:5):

- "Share partially complete design information
- Organise for direct, worker-to-worker collaboration
- Develop a sense of when decisions must be made
- Develop a sense of how to absorb changes
  - Avoid repetition
  - Separate concerns
  - Encapsulate variation
  - Defer implementation of future capabilities
- Commit to refactoring

- Use automated test suites"

Tools: Option thinking, the last responsible moment, making decisions (Steindl, 2004; Norton, 2005).

4. *Deliver fast*

The goal here is to create value as fast as possible, once the customer decided what he/she wants. This means (Poppendieck, 2003:5):

- no delay in deciding which requests to approve
- no delay in staffing and immediate clarification of requirements
- no time consuming handoffs
- no delay in testing
- no delay in integration
- no delay in deployment

In mature software development organisations "all of this happens in one smooth, rapid flow" in response to customer requirements (Poppendieck, 2003:5).

Tools: Pull system, queuing theory, cost of delay (Steindl, 2004; Norton, 2005).

5. *Empower the team*

Using LD as an SDM, things are done fast and fast decisions should be made by people doing the work. In LD, the team makes its own process designs, commitments, goals and decisions on how to complete these specified goals. A team can only be allowed to make its own decisions if it is empowered through expertise, training and leadership.

Once a team is empowered, it can make better decisions. It is management's responsibility to supply the project teams with the necessary

training, expertise and leadership as well as information to make the best decisions in order to deliver a successful project. Working directly with customers in order to understand their requirements, and working with other developers to figure out how these requirements could be satisfied, results can be presented frequently to customers to determine whether development processes are on track.

Tools: Self-determination, motivation, leadership, expertise (Steindl, 2004; Norton, 2005).

6. *Build integrity in*

Two kinds of integrity exist, namely perceived integrity and conceptual integrity. Perceived integrity is exactly what the customer wanted, although he/she did not ask for it. In order to achieve perceived integrity, the organisation should have a continuous and detailed flow of information from the users to the developers. This is achieved where the master designer (architect) understands the detail of the domain and ensures that developers always have user requirements at hand to make correct decisions that will be of value to customers.

Conceptual integrity presents software to customers with a single metaphor of how the tasks are completed to satisfy requirements. "Conceptual integrity means that all parts of a software system work together to achieve a smooth, well functioning whole." (Poppendieck, 2003:6). The flow of detail information between team members and technical members of a project will achieve conceptual integrity within a balanced system. Everyone, from supplier to customer, should be involved in the progress of development from the beginning of the project. Many believe that integrity comes from a documentation-centric approach, but according to Poppendieck (2003:7), organisations should rather use a test-centric approach. The author further states that organisations should "test early, test often, test exhaustfully, and

make sure an automated test suite is delivered as part of the product."

Tools: Perceived integrity, conceptual integrity, refactoring, testing (Steindl, 2004; Norton, 2005).

7. *See the whole*

   Overall success of the project is more important to LD than the traditional sub-optimisation of individual tasks. The most appropriate way to encourage collaboration and avoid sub-optimisation is to make the team accountable for their control and influence. This means measuring the team's performance and defect count, and not that of the individual team member. It may seem unfair to hold the whole team accountable for an individual team member's performance but this cause teams to work together, sort themselves out, and take responsibility to plan their own processes.

   Tools: Measurements, contracts (Steindl, 2004; Norton, 2005).

According to Highsmith (2002a:6), Bob Charette's LD sends three messages to developers using ASDMs:

- "The wide adoption of ASDEs (Agile Development Software Ecosystems) will require strategic selling at senior levels within organisations
- The strategic message that will sell ASDE's is the ability to pluck opportunity from fast-moving, high-risk exploration situations
- Proponents of ASDEs must understand and communicate to their customers the risks associated with agile approaches and, therefore, the situations in which they are and are not appropriate"

Maturity (*deliver fast*) is measured by operational excellence, i.e. the speed with which customers can be served repeatedly and reliably (Poppendieck, 2003:3). Maturity is not measured by the manner in which plans are followed or the

"comprehensiveness of process documentation" (Poppendieck, 2003:3).

These lean principles identified by Poppendieck (2003:3-7) have lead to extraordinary improvements in several areas such as healthcare delivery, logistics, building construction, product development and military logistics. Highsmith (2002a:6) also states that LD has been successful in a number of large telecommunication projects in Europe.

2.5 The effectiveness of ASDMs

ASDMs are gaining popularity and many organisations are adopting ASDMs since the creation of ASDMs in the early 1990's (Good, 2003:27). The questions asked by organisations and developers are: Do ASDMs really work? How effective are these ASDMs? Do they work in all circumstances or only in specified circumstances?

In general, there is little information about the effectiveness of some of the settled ASDMs explained in this literature study. Some of them are only tested by the authors themselves, but a methodology truly works when other developers (not the authors) test it and find it to be of value. In recent years, there have been more intensive analysis of the effectiveness of ASDMs, but gathering hard empirical data about ASDMs are difficult because of the nature of software production (Good, 2003:29).

Organisations do not actually specify if the ASDMs they used, have failed. The first question that should be asked is: Was the methodology chosen used correctly according to the specifications (e.g. principles, practices, rules, guidelines)? If not, organisations and role-players cannot say the chosen ASDMs do not work.

Ironically, developers always hear success stories about ASDMs. Where are the bad and sad stories of failure? Some have been documented, but most are

not even mentioned. ASDMs have some "bad smells", which can only be identified by people using these methodologies (excluding the authors), who will test them in different environments. There is no perfect methodology that will work in all kinds of environments and under any circumstances. Many of these ASDMs work only for small teams and projects while others were proven successful in both small and large projects.

Because of research, many lessons have been learned about ASDMs. However, more success stories about small projects have been documented, and fewer about large projects (Lindvall *et al.*, 2002:206; Cohen *et al.*, 2003:31). This could be because managing a large project is much more difficult. Team size does not matter in the most ASDMs, although communication in larger teams are more complex.

Lindvall *et al.* (2002:206) explain some lessons learned from using ASDMs in organisations:

- Experience in agile projects is very important for it to succeed, although experience in the actual building of the system is more important. It is estimated that 25% - 33% of project personal must be experienced, but in cases where pair programming is practiced in teams where they monitor each other, it might be as low as 10%.
- ASDMs can be used to conduct safe-critical and reliable projects. Critical issues are easily addressed in ASDMs, because customers give requirements, state the importance of each requirement and provide input through system development. The key is that the performance requirements are made explicit early, and that proper levels of testing are planned.
- ASDMs need less formal training than traditional SDMs. Training is minimized by the fact that pair programming is used where team members monitor each other. This is more important than regular training, because of the experience gained from learning from one another. There is training material available for XP, Scrum, FDD and Crystal ASDMs.

- The most important success factors are culture, people and communication. ASDMs need cultural support otherwise the methodology applied will not succeed. ASDMs use fewer but more competent people than traditional SDMs. Communication is enhanced by using pair programming, and constant interaction with customers who give frequent feedback.
- Using ASDMs in projects, warning signs can be detected early in project development. Warning signs include low interest in meetings and production of "useless documentation".
- Refactoring should be done on a regular basis with code of reasonable size, keeping the scope down and local. ASDMs make large scale refactoring more feasible than traditional SDMs. If a set of automated tests is maintained, changes to big architectural designs do not have to be risky.
- Documentation makes the design "heavy" and should be assigned as a cost. Organisations normally ask for more than is needed. In order to give value and satisfy requirements, the main goal should be communication, and useless documentation should be avoided.

In 2002, Reifer (2002:16-17) surveyed 32 development organisations of which fourteen were using ASDMs on 31 individual projects. The result of his study was that seven of the fourteen organisations that used ASDMs captured hard cost, productivity and quality data. Five of the seven had benchmarks that they could use for comparison purposes (Reifer, 2002:17). Reifer (2002:17) came to the following conclusion about organisations using ASDMs:

- *Productivity Improvement:* 15% to 25% average gain based on published industry benchmarks.
- *Cost Reduction:* 5% to 7% on average based on published industry benchmarks.
- *Time-to-market compression:* 25% to 50% less time compared to previous projects in participating firms.
- *Quality improvement:* Five firms had data showing that their defects rates were on par with their other projects when products or applications were

released.

Good (2003:27-28) documented another global survey of experience using ASDMs carried out by an Australian company. The results of this study are summarised as follows:

- 88% of organisations cited improved productivity
- 84% reported improved quality of software production
- 46% of respondents reported that development costs were unchanged using ASDMs, while 49% stated that costs were reduced or significantly reduced
- 83% stated that business satisfaction was higher or significantly higher
- 48% cited that the most positive feature of agile methodologies was their ability to respond to change rather than follow a predefined plan

Cockburn and Highsmith (2001b:133) quotes a survey of ASDMs and rigorous methodologies conducted by the Cutter Consortium in 2001 to which nearly 200 people from a wide range of organisations in North America, Australia, India, Europe and other locations responded. This survey showed the following results (Cockburn & Highsmith, 2001b:133):

- Many organisations said that they were using at least one ASDM.
- ASDMs show better delivery performance than rigorous methodologies in terms of quality, client satisfaction and business performance.
- ASDMs scored better in terms of employee morale than rigorous methodologies (54% of respondents were IT and executive managers and only 12% were developers).

An electronic workshop was held by Scott Ambler (independent consultant specializing in object-oriented development), Dr Barry Boehm (well known for his spiral software development lifecycle and COCOMO II estimating technique), Kent Beck (originator of XP), Alistair Cockburn (author of *Agile Software Development*) and Randy Miller (co-author of *Advanced Use Case*

Modelling). The workshop focused on three points, and they were estimated to be true for ASDMs (Ambler, 2002:9):

1. Agile development works better for smaller teams (up to 20 or 30 members).
2. ASDMs work best when the future is unknown (designed for current, not future needs).
3. ASDMs fit applications that can be built quickly and do not require extensive quality assurance. ASDMs work less well for critical, reliable, and safe systems.

Even the United States Department of Defence specifies that ASDMs will be used in their software development (Good, 2003:29). Ambler (2002:9) states that "agile processes are 'real': they're here to stay and every IT professional needs to take them seriously".

These results shows a very positive attitude towards ASDMs and many significant software development companies wanted to implement these ASDMs (Good, 2003:28). Companies started to utilise ASDMs, and a number of large software customers demanded their software must be developed using ASDMs.

These results and success stories are evidence that supports the conclusion that ASDMs deliver software of business value to the customer, on time and within budget, if implemented correctly.

2.6 Summary

In this chapter, the researcher firstly identified an SDM using four main aspects identified by Huisman and Iivari (2006:32). According to these aspects, an SDM consists of a:

- System development approach(s)
- System development process model(s)
- System development method(s)

- System development technique(s)

After defining an SDM, the researcher could define an ASDM and explain the seven core ASDMs (Highsmith, 2002a:6) that are relatively settled in today's environment. In order for an organisation to be agile, Highsmith (2002b) explains agility as the ability to deliver quickly, change quickly, and to change as often as possible. Lindvall *et al.* (2002:201) explains ASDMs as iterative, incremental, self-organizing and emergent methodologies.

During the explanation of the seven ASDMs, it became clear that ASDMs focus on people, incremental development and communication. A summary of the seven core set ASDMs are given in Table 2.1.

Lastly, the researcher studied the effectiveness of ASDMs used in practice today. ASDMs give developers the ability to adapt to an ever-changing environment to produce a product that is of value and up to date. There are many success stories of organisations that used ASDMs in their projects, but there are circumstances where ASDMs are not that successful. There is no unique SDM that will work in all circumstances. This also applies to ASDMs. Some ASDMs work only in small projects while others work only in large projects. Furthermore, there are ASDMs that might work in both large and small projects. The study of the effectiveness of ASDMs is still taking place, but the fact that they are being used, has been established.

Highsmith (2002a:9) states that large companies in countries like the USA, Australia, Europe and India found ASDMs to be very effective, because a product could be produced in a "turbulent, ever-changing, ever-existing marketplace". Ambler (2002:9) agrees with Highsmith (2002a:9) that "agile processes are 'real': they're here to stay and every IT professional needs to take them seriously".

The evidence supports the conclusion that ASDMs can deliver software on

time, within budget, and in a constant changing environment if implemented correctly.

Name:	Time and Author:	Core Ideas:	Scope of use:	Current status
XP	Created by Kent Beck in 1996 and later popularized in his book in 1999.	Twelve key practices (such as refactoring, test before coding). No process to fit every project.	Good for small and medium size teams of 3-20 members	Growing. More practical experience than academic research.
Crystal	Developed by Alistair Cockburn in 1999.	A set of methodologies. Suggest development cycle within four months. Emphasis on communication. Allow adoption of other ASDMs.	Not good for life-critical system. Up to 40 persons in local development.	Four proposed Crystal ASDMs, two of them exist.
Scrum	Developed by Ken Schwaber and Jeff Sutherland in 1995.	Do not require specific practices, but need management practices and tools.	Suitable for small teams: <10 members.	Ongoing research aims to integrate Scrum and XP.
DSDM	Defined by the 16 founding members of the DSDM Consortium in 1994.	Applications of controls to RAD. Emphasises time and resources.	Team size between 2 and 6, multiple teams exist. Can be used in large system, if the system can be split into components.	Widely used in UK. E-DSDM was released in 2001.
ASD	Created by Jim Highsmith and first documented in his book in 2000.	Emphasis on incremental, iterative development.	Focuses on developing large system. No built-in limitation.	No significant research.
FDD	Created by Jeff De Luca and Peter Coad in 1997	Focuses on design and building phase. Emphasises iterative development. Needs other supporting approaches.	Claims to be suitable for the development of large software project.	Some consulting firms advocating it. Relatively new and still evolving.
LD	Created by Bob Carette during 1980's, later popularized by Poppendiecks book in 2003.	7 Principles used as guideposts with 22 tools.	Highsmith (2002a:6) states LD is effective in large projects.	Used in large telecommunication, health, logistics, military and construction projects (Poppendieck, 2003:3).

Table 2.1: Summary of ASDMs (extracted from Abrahamsson et al., 2002:18-73).

CHAPTER 3

DATA WAREHOUSING

3.1 Introduction

A number of approaches for developing a data warehouse exist, but for the purpose of this study, the researcher will only explain the approach of Bill Inmon, the father of the data warehouse; as well as that of Ralph Kimball, the dimensional data warehouse expert.

In this chapter, business intelligence (BI) will firstly be explained after which a data warehouse will be defined by examining the different definitions offered by various authors. The approaches of Inmon and Kimball will then be discussed, starting with the explanation of data flow through Kimball's high-level technical architecture components. In contrast, Inmon's hub-and-spoke architecture will be explained in par. 3.6.1. Each architecture will be followed by a discussion on both expert's data warehouse development lifecycles, using the four main stages (requirements collection, data modelling, data staging, and data access and deployment). The chapter will end with a description of the contrasts and differences in the approaches of Kimball and Inmon.

3.2 Business intelligence

The term business intelligence (BI) was used as early as 1989 by Howard Dresner from the Gartner Group who "popularized BI as the umbrella term to describe a set of concepts and methods to improve business decision-making by using fact-based support systems" (Wikipedia, 2006). According to Greiner (2001:13), a Gartner Group report concluded in 1996 that by the year 2000, "Information Democracy will emerge in forward-thinking enterprises, with business intelligence information and applications available broadly to employees, consultants, customers, suppliers, and the public".

BI is the tracking, collection, understanding, management, and analysing process of gathering information about an organisation's competitors and environment (Wikipedia, 2006; McGuigan, 2006). BI is all about gathering data that the organisation already generates and organizing it into information that is of value to business growth and the prediction of future events. According to TechTarget (2005), "BI is a broad category of applications and technologies for gathering, sorting, analysing and providing access to data to help enterprise users make better business decisions. BI applications include the activities of decision support systems, query and reporting, on-line analytical processing (OLAP), statistical analysis, forecasting and data mining". Through better understanding the data the organisation generates, it can determine where changes can be made to help make the business more efficient, increase revenue, decrease costs, and improve relationships with customers and suppliers.

Once an organisation has a BI system in place, it has a good understanding of how to compete with the strongest competitors, improve turnaround times on data collection, more targeted marketing campaigns and a better understanding of customer needs. BI improves the organisation's agility (i.e. ability to adapt to change) in order to take better advantage of constant evolving environmental conditions (McGuigan, 2006).

BI results in better decision-making by transforming a large amount of data into relevant and accurate information that will be of value. "BI encompasses the gathering, sorting and analyzing of data," because it includes tools in various categories. Example categories include customer relationship management (CRM), data warehousing, decision support systems (DSS), forecasting and online analytical processing (OLAP) (Government Technology, 2001).

BI components framework

The framework consists of three layers, namely the business layer,

administration and operation layer as well as the implementation layer.

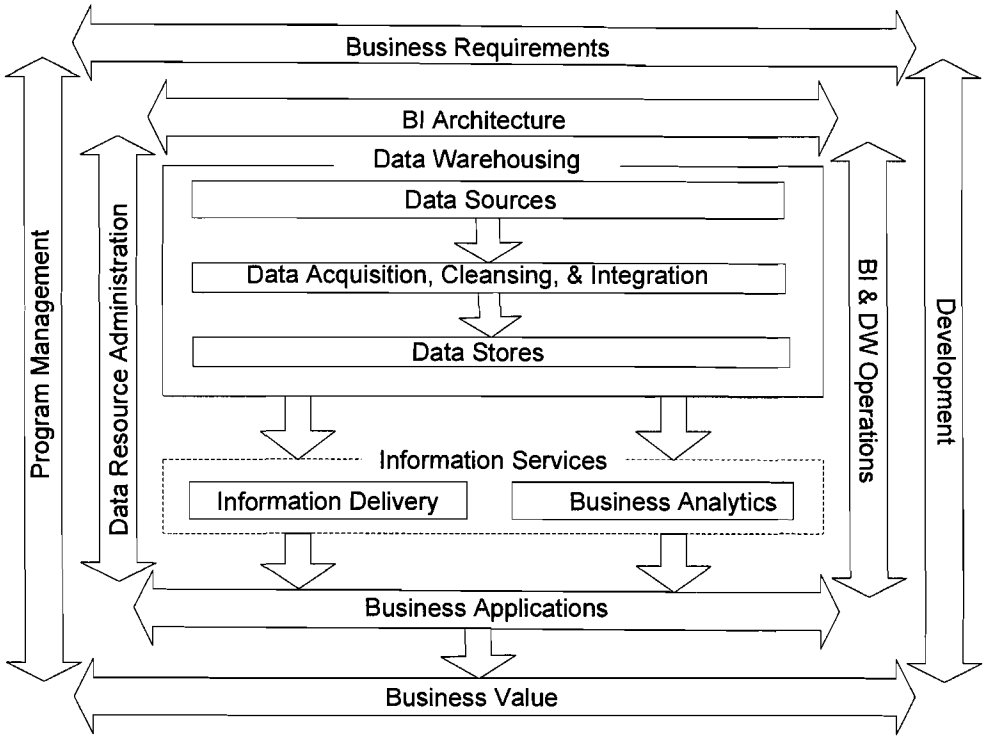


Figure 3.1: BI components framework (TDWI, 2004:10)

The business layer consists of the components required for BI to fit into business processes, activities and organisations. The components include (TDWI, 2004:11):

- *Business requirements:* The results expected as well as the reason to implement BI.
- *Business value:* The benefits of implementing BI, including reduction costs, improved profit margins and increased revenue.
- *Program management:* Ongoing activity of managing the implemented BI program to gain maximum business value.
- *Development:* Project activities that develop and deploy data warehouse and BI products, including project decomposition and methodologies.

The administration and operation layer consist of the components that connect business components with technical components. The layer is composed of

(TDWI, 2004:11):

- *BI architecture:* Conventions, standards and frameworks that describe BI environment components and the relationships among them including project, organisational, technology, business and data architectures.
- *Business applications:* Archiving business results through the use of business processes and procedures that access and/or retrieve and employ information.
- *Data resource administration:* Policies, processes and procedures to govern data.
- *BI & Data warehouse operations:* Executing, maintaining and monitoring acceptable availability, performance and quality of data warehousing and BI functions and services.

The implementation layer consists of the technical components required to capture data, convert it into information and present the information to the business. This layer consists of (TDWI, 2004:11):

- *Data warehousing:* Process to integrate data and prepare it to become information. A detailed definition is given in par. 3.3.
- *Information services:* Processes, procedures and systems that turn data into information and deliver the information to the business.

Deployment of BI

Microsoft describes the deployment process of BI starting with the importance of implementing proper analysis, design and planning procedures before deployment (Government Technologies, 2001). After implementing these procedures, a sound architectural model is developed in which business process views aught to be created from the data the organisation has already collected. This data should then be analysed so the information generated can be integrated with information of data sources.

Deployment can begin after a successful architectural model is developed. A

BI-system should contain the five elements including a database, an ETL (extract, transform, load) function, analytic tools, reporting/querying tools, as well as user training. Microsoft warns organisations not to try to do too much at once, as BI is a simple and inexpensive way to analyse years of data.

The best productivity gain that BI offers is the ability to access data faster, and the most effective driver behind BI investments is better customer satisfaction and retention (Macling, 2004:20). A real BI system connects workers and customers with the right information at the right time (Windley, 2003:44) and gives them all the information they need. Many organisations have achieved great success in using and implementing BI and BI tools (Windley, 2003:44; Macling, 2004:20). Data warehousing is used as a BI tool. Because BI uses an incremental approach to identify requirements and issues that cause problems for an organisation (Government Technologies, 2001), it will be a good idea to implement BI, or a BI tool such as a data warehouse, using methodologies that focus on incremental development and deployment, i.e. ASDMs.

3.3 What is a data warehouse?

There is no official standardised definition for a data warehouse supported by a standards committee such as the American National Standards Institute (ANSI) (Quarles, 2002:6). Bill Inmon (1996), the father of data warehousing, and the dimensional data warehousing expert Ralph Kimball (Kimball *et al.*, 1998) have different explanations towards the development and definition of a data warehouse.

Kimball defines a data warehouse as "the queryble source of data in the enterprise" (Kimball *et al.*, 1998:19). Kimball also states that a data warehouse is the "union of its constituent data marts" (Kimball *et al.*, 1998:27).

Inmon (1996:371) defines a data warehouse as "a collection of integrated, subject-oriented databases designed to support the DSS function, where each

unit of data is specific to some moment of time. The data warehouse contains atomic data and lightly summarized data".

Inmon (1996:33) further defines a data warehouse as a "*subject oriented, integrated, non-volatile,* and *time variant* collection of data in support of management's decisions. Each part of this definition can be explained as follows (Hou *et al.*, 1998:2-3; Inmon, 2000:1-7; McKnight, 2005):

- *Subject oriented*: A data warehouse is oriented around the major subjects of the enterprise. Operational data is organised around business activities or functional areas. Subject orientation presents data in a format that is much cleaner and more consistent for users to understand, and focuses on natural data groups.
- *Integrated*: Data integration within the data warehouse is the most important. Data integration is accomplished by dedicating consistency in naming conventions, measurement of variables, encoding structures, and physical attributes of data.
- *Time variant*: Data in the data warehouse is accurate as of some moment in time, because a data warehouse maintains both historical and current data.
- *Non-volatile*: Only load and access operations are allowed. The data in the data warehouse is never updated like in an operational environment.

Meyer and Cannon (1998:6) state that the primary goal of a data warehouse is the creation of a single, logical view of an enterprise's data, accessible by developers and business users alike. Goede (2005:133) defines a data warehouse as examples of decision support systems (DSS).

According to TDWI (2004:24), "A data warehouse is a data structure that is optimised for distribution. It collects and stores integrated sets of historical data from multiple operational systems and feeds them to one or more data marts. It may also provide end-user access to support enterprise views of data."

There is a broad category of definitions for DSS, but in data warehousing terms, a DSS can be described as a computer based system that aids workers with the correct information to make informed decisions during the decision making process (Gachet, 2000:1-2; Gachet & Haettenschwiler, 2003:142).

Because there is no universally accepted definition of a data warehouse, the popular search engine Google (http://www.google.co.za) shows a wide variety of descriptions and definitions. Many state that it is a collection of databases or data that can be loaded, extracted and transformed, while others emphasise that it is more of an information structure or data repository that preserves historical and current data to be accessed and analysed to create reports.

3.4 Definitions associated with data warehousing

After understanding what a data warehouse is, definitions associated with the concept can be defined. The researcher will only define definitions associated with the approaches of Inmon (1996) and Kimball *et al.* (1998).

Dimensional modelling: It is a logical design technique with a primary objective of presenting data in a standardised framework, with high performance access inside a data warehouse. Furthermore, it is a name for a logical design technique in data warehousing (Kimball *et al.*, 1998:140,144), and an alternative for the term ERD-modelling.

Fact: A fact is something that you did not know before, normally a numeric value that changes over time (Kimball *et al.*, 1998:165). A fact has to do with the transaction that takes place, e.g. price, quantity, VAT.

Additive facts: These are facts that can be added along all the dimensions (Kimball *et al.*, 1998:193), e.g. quantity.

Semi-additive facts: These facts are snapshots of a specific point in time; they do not present a flow past this snapshot (Kimball *et al.*, 1998:193), e.g. bank statement.

Non-additive fact: Facts that have no meaning; if an average is not

computed, e.g. room temperature.

Fact table: A fact table is the primary table in a dimensional model. It contains all the primary keys of the dimension tables as foreign keys. These foreign keys form a multi-part key that uniquely identifies the table. The fact table also contains one or more facts. The fact table is the primary table in "each dimensional model that is meant to contain measurements of the business" (Kimball *et al.*, 1998:17,144,165).

Dimension table: The dimension tables are arranged around the fact table in the star-schema and they contain attributes that describe the characteristics of a record. They also contain a primary key that is related to its corresponding foreign key in the fact table. A dimension table is "one of a set of companion tables to a fact table" (Kimball *et al.*, 1998:17,144,166).

Business process: It is a set of business activities that are of value to the business users of the data warehouse. It is a useable set (grouping) of information that is represented by a star-schema (Kimball *et al.*, 1998:18,348).

Data mart: A data mart can be organized around a single business process. A dimensional model, with its fact table and corresponding dimension tables, represents a data mart. A data mart contains granular data and in some cases it also contains aggregates. It is a "logical sub-set of the complete data warehouse" (Kimball *et al.*, 1998:18, 27,348), meaning that a collection of data marts form a data warehouse.

OLAP (on-line analytical processing): Kimball *et al.* (1998:21) defines OLAP as "the general activity of querying and presenting text and number data from data warehouses, as well as a specifically dimensional style of querying and presenting that is exemplified by a number of OLAP vendors". OLAP is a form of transaction processing conducted via a computer network where the response time is crucial for business success (Inmon, 1996:30).

ROLAP (relational OLAP): "A set of user interfaces and applications that give a relational database a dimensional flavour" (Kimball *et al.*, 1998:21).

MOLAP (multi-dimensional OLAP): "A set of user interfaces, applications, and proprietary database technology that have a strong dimensional flavour" (Kimball *et al.*, 1998:21).

Metadata: Metadata is data about data, in other words it is data that explains other data. It is information in the data warehouse other than the actual data. (Kimball *et al.*, 1998:22,435), e.g. column headings in a table.

Snowflaking: Snowflaking takes place where a table can be divided into additional tables. It is done by taking low cardinality text attributes from a dimension table and placing them in a secondary dimension table. The secondary table is uniquely related to the primary table. In other words, an attribute has the ability to create an additional secondary table, giving the impression of a snowflake effect. (Kimball *et al.*, 1998:151).

Stovepipe data mart: Stovepipe data mart does not use conformed dimensions. In other words, it does not use dimensions of other data marts or dimensional models (Kimball *et al.*, 1998:18).

Primary key: Defines uniqueness in a dimensional table within a dimensional model (Kimball *et al.*, 1998:191). The attribute that uniquely identifies a record in a table is called a primary key.

Surrogate key: The key is normally an integer number (1, 2, 3...). The key alone does not mean anything and does not have any value (Kimball *et al.*, 1998:192).

Foreign key: This key is normally in the fact table. Each foreign key in the fact table relates to its corresponding dimension primary key (Kimball *et al.*, 1998:191).

Conformed dimension table: It is when a dimension table that is related to one fact table in a dimensional model connects to another fact table in a different dimensional model (Kimball *et al.*, 1998:157). Example: client dimension or product dimension.

Conformed fact: Conformed facts are required when the same terminology is used across data marts and when single reports are built when drilling across data marts (Kimball *et al.*, 1998:159). Example: revenue, costs and profit.

Factless fact table: Is a fact table set up without any facts. Example: an event that takes place when you want to compute the attendance of students at college (Kimball *et al.*, 1998:212).

Aggregate: An aggregate is a summary of data that is already in the

dimensional model, built to improve query performance so fast and effective queries can be done. An aggregate is the result of a big query, stored as a table (Kimball *et al.*, 1998:211,383,647).

Star-schema: A star-schema represents a dimensional model. It consists of a primary fact table and its corresponding dimensions arranged around the fact table. The star-schema is easy to understand and changes can be made easily. (Kimball *et al.*, 1998:206-211,589).

Granularity: Grain is the depth of detail. The depth of detail can be declared by using "per" (Kimball *et al.*, 1998:195). Granularity refers to the level of detail, meaning the less detail there is, the higher the level of granularity (Inmon, 1996:45,373). Example: products per client per day.

Integrity: The data in the data warehouse should be as accurate and consistent as possible (Inmon, 1996:374).

Redundancy: The same data is stored more than once. "Practice of storing more than one occurrence of data" (Inmon, 1996:377).

Aggregate navigator: Aggregate navigator is a component that gives us the awareness of aggregates. It is the piece of middleware that sits between the client and the database management system (DBMS). The aggregate navigator sits above the DBMS (database management system) and catches all SQL (structured query language) statements and queries sent by the user. A good aggregate navigator tool holds statistics about every SQL query as well as the use of the current aggregates. It also mentions aggregates that should be built in order to improve performance (Kimball *et al.*, 1998:383)

Data staging: Data staging is also known as the ETL (extract, transform and load) process. Data staging is the process of cleaning data so it can be of value to the data warehouse (Kimball *et al.*, 1998:23).

Architecture: Architecture gives better process planning and communication in the project. The flexibility and productivity will improve with an effective architecture. Architecture is like the blueprint of a house, you can see the value and purpose of the architecture. Example: an architect and client can decide what the results should be. The contractor computes the price and time of the project, while sub-contractors see where they fit in and what work they should

do (Kimball *et al.*, 1998:318).

Data mining: There is a wide variety of definitions for the term data mining because it is used in several fields of study. A summary definition can be created using Friedman (1998:3). Data mining is the process of identifying and extracting previously unknown and potential useful patterns in data, as well as distinguishing previously unknown relationships within the data to make crucial business decisions. Data mining is the process by which an organisation compiles personal, pertinent, actionable information about the purchasing habits of their current and potential customers (Straubhaar & La Rose, 2003:338). According to Kimball *et al.* (1998:377), data mining is a "collection of powerful analysis techniques for making sense out of very large data sets".

Multi-dimensional database cube: A multi-dimensional database (MDD) is a specialized engine that stores data in a proprietary format (also known as MOLAP) that is commonly referred to as a cube. The cube corresponds to business dimensions understood by users (Kimball *et al.*, 1998:408).

ODS (operational data store): According to Inmon (1995:21), an ODS "is a subject-oriented, integrated, volatile, current valued, detailed-only collection in support of an organisation's need for up-to-the-second, operational, integrated, collective information". Example: large companies with many bank accounts such as AT&T, are managed by the bank through creating an ODS for AT&T as a single account (Inmon, 1995:21).

DSS (decision support system): According to Inmon (1996:372), DSS is "a system used to support managerial decisions". DSS involves the analysis of data and as a rule, it does not involve an update process.

ERD (entity relationship diagram): An ERD-diagram shows the relationship among entities. According to Inmon (1996:373), an ERD is the schematic presentation of "all entities within the scope of integration and the direct relationship between those entities".

3.5 Kimball's approach towards data warehouse development

Kimball and Inmon's approaches towards the development of a data

warehouse will be explained at the hand of their books and interpretations.

3.5.1 High-level technical architecture

The data flow of the high-level technical architecture model can be seen in the following figure:

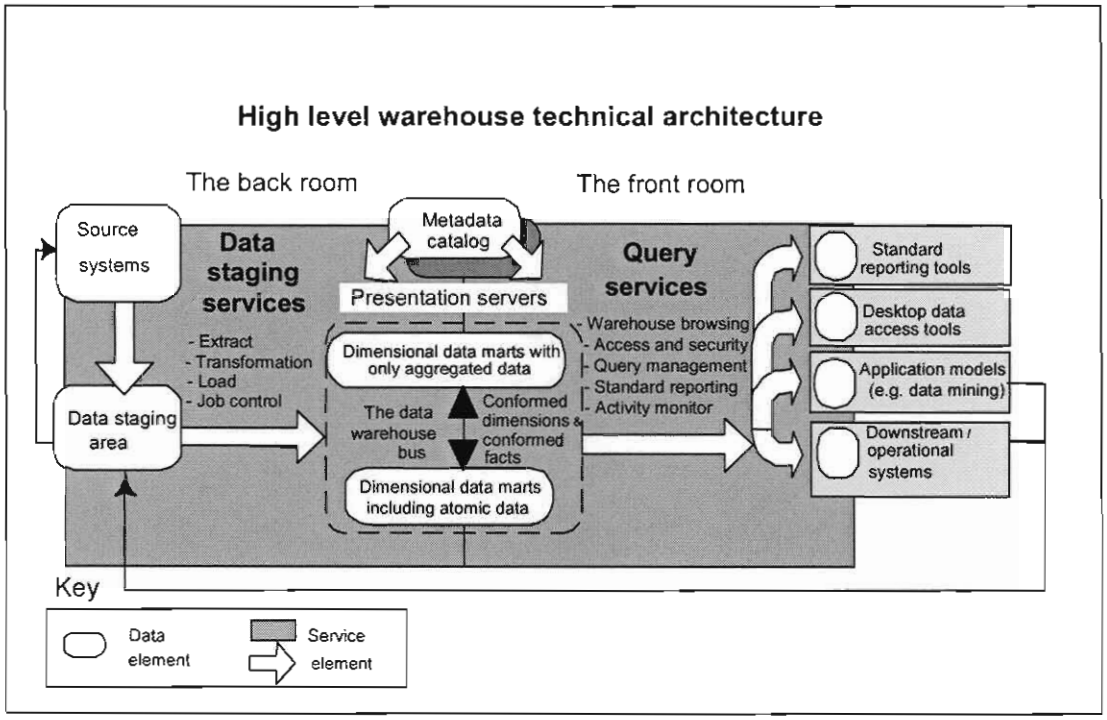


Figure 3.2: The high-level technical architecture (Kimball et al., 1998:329).

The data element, services and elements will be explained in detail in the paragraphs that follow: Each data element and service will be defined, as well as the data flow through the back and front rooms. The data flow can be seen in figure 3.2 by following the arrows.

The back room: This is the place where the data staging process takes place. It is described as the engine room of the data warehouse and the primary concern is to write data, with appropriate associations and time, from point A to B. (Kimball *et al.*, 1998:335).

The front room: The front room is the public face of the data warehouse

(Kimball *et al.*, 1998:373) that users see and work with every day. Users do not know what is going on behind the user interface; the primary goal should be to make information as accessible as possible.

Data stores: Are the permanent or temporary landing places for data along the way in the technical architecture (Kimball *et al.*, 1998:330).

According to Kimball *et al.* (1998:16) three different systems are required for a data warehouse to function successfully, namely the *source system*, the *data staging area*, and the *presentation server*.

Source system: It is an "operational system of record whose function is to capture the transactions of the business" (Kimball *et al.*, 1998:14). The source system is outside the data warehouse and uses keys to uniquely identify data (Kimball *et al.*, 1998:16). It does not contain any built in dimensions. The data warehouse requests access to the source system, which the source system then grants, with strict access rules (Kimball *et al.*, 1998:371). The data is read from the source data and extracted to the data staging area using data staging services (Kimball *et al.*, 1998:436).

Data staging area: The data staging area is "a storage area and a set of processes that transform, clean, combine, duplicate, household archive and prepares source data for the use in the data warehouse" (Kimball *et al.*, 1998:16). This is the construction site of the data warehouse. It is everything between the source system and presentation server. Here data transformation takes place and value is added to the data. The data staging area is normally spread over a few machines and does not have any query or presentation services. It is not meant to be seen by users. (Kimball *et al.*, 1998:16). After the data has been cleaned it is moved to the presentation server.

Presentation server: "... is the target physical machine on which the data of the data warehouse is stored and organised for direct querying by end users, report writers and other applications" (Kimball *et al.*, 1998:16). Data should be in a dimensional framework.

Dimensional data marts including atomic data: "Atomic data marts hold data at

the lowest common-denominator level" (Kimball *et al.*, 1998:347). This means data is held at the lowest level of detail (atomic data), which cannot be divided into smaller pieces, in order to meet business requirements. Atomic data marts may contain a range of aggregates to improve performance (Kimball *et al.*, 1998:347).

Dimensional data marts with only aggregated data: Data that is related to every core business process is called a business process data mart (Kimball *et al.*, 1998:348). Every core business process generates data that can be used and is of great importance for other business functions. Business process data marts bring relevant sets of data together from the atomic data mart to present it in a simple dimensional form which users can understand and that has meaning and is important to the users (Kimball *et al.*, 1998:348).

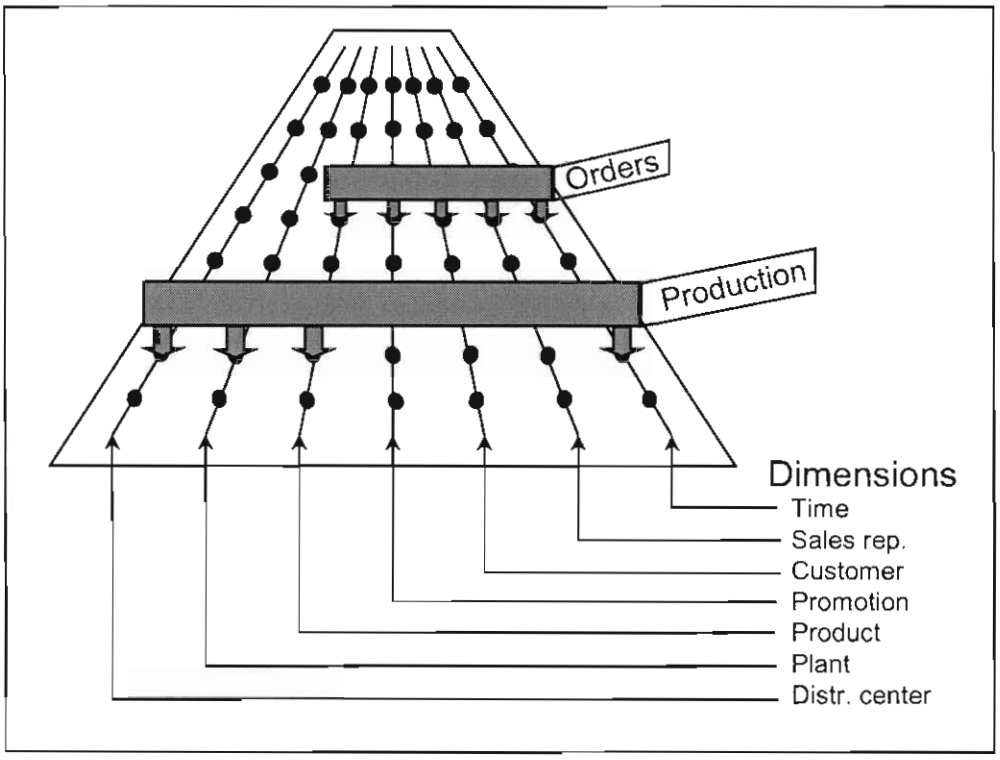


Figure 3.3: The data warehouse bus (Kimball et al., 1998:347)

According to Kimball *et al.* (1998:346), in figure 3.3, every dimension is seen as a connector (wire). Each of the business processes is seen as an expansion card that plugs into the appropriate data connectors. Figure 3.3 gives us an

example of how the bus might work. In this example *time*, *sales rep*, *customer*, *promotion*, *product*, *plant* and *distr. centre* are dimension tables. Each of these tables are seen as a wire that can be connected to more than one data mart (orders, production) to form conformed facts and conformed dimensions (see par.3.4).

"Services are the functions required to accomplish the required tasks of the data warehouse" (Kimball *et al.*, 1998:330). For tasks that should be completed, there are data staging services; functions that can clean the data, and query services; functions that allow easy access and valuable feedback.

Data staging services: The data staging services are the tools and techniques that must be used in the data staging process (Kimball *et al.*, 1998:350).

- *Extract:* Tools and techniques are used to pull data from the source system. The biggest challenge is to determine which data should be extracted and which kind of filters should be applied (Kimball *et al.*, 1998:357). It is important to understand the requirements of the extraction process in order to determine which kind of services will be required. Most extract processes generate temporary load files that become the input data for the next activity downstream (Kimball *et al.*, 1998:357). Mayor classes of requirements include multiple sources, code generation, multiple extract types, replication and compression/decompression (Kimball *et al.*, 1998:358-360)
- *Transformations:* Data transformation takes place when the data is extracted from the source system. A range of acts is performed on the data to change, edit and convert it in order to present it in an acceptable format to users, so it can be of value to the business. (Kimball *et al.*, 1998:361). Some kinds of transformation that might be necessary in the data warehouse include integration, denormalisation and renormalisation, cleaning, merging/purging, data type conversion, calculation, derivation, allocation and aggregation (Kimball *et al.*, 1998:360-363).
- *Loading:* The loading process should support as many targets as possible.

Tools should be used to optimise the load process using the features provided by a bulk loader or using incremental load processes. After loading, it is important to have services that will support requirements, like creating or dropping tables or indexes. Capabilities necessary during the loading process may include support for multiple targets, load optimisation and entire load process support (Kimball *et al.*, 1998:363-364).

- *Job control:* "The entire data staging job stream should be managed" by using a single metadata-driven job control environment (Kimball *et al.*, 1998:364). The job control service also captures metadata of the job itself as well as metadata that has to do with statistics. It is important to develop an environment that creates, manages and monitors the job stream of the data staging process. (Kimball *et al.*, 1998:364). Job control services include job definition, job scheduling, monitoring, logging, exception handling, error handling and notification (Kimball *et al.*, 1998:364-366).

Query services: The data access services should stand alone, without being dependant on specific tools. They should be available to all and should add value to the data access process.

- *Warehouse browsing:* Warehouse browsing uses the "metadata catalogue to support user efforts in order to find and access the required information they need" (Kimball *et al.*, 1998:379). The *metadata catalog* provides information and parameters that "allow the application to perform their tasks – a set of control information about the data warehouse, its contents, the source systems, and its load processes" (Kimball *et al.*,1998:332).
- *Access and security:* "Access and security services facilitate a user's connection to the database" (Kimball *et al.*, 1998:380). It relies on the authentication and authorisation (i.e. a person really is whom he claims to be) services where access rights are determined or access is refused to an identified user. It will be a sensible option to give every user a unique identification number (Kimball *et al.*, 1998:380). Authentication can be established by a password as well as physical evidence such as a

fingerprint or retina scan.

- *Activity monitor:* This service captures information about the use of the data warehouse. The activity monitoring service should centre around four areas, namely performance, user support, marketing and planning (Kimball *et al.*,1998:380-381).
- *Query management:* According to Kimball *et al.* (1998:381), query management services are the set of capabilities that manage the exchange between the query information, execution of the query and the return of the result set to the authorised user's desktop. Here the user interacts with the data warehouse to access required information. The mayor query services that should be included in a data warehouse include content simplification, query reformulation, query retargeting and multipass SQL, aggregate awareness, date awareness and query governing.
- *Standard reporting services:* Here standard reports can be created with a production style fixed-format that has a limited lifespan. These reports are normally displayed to a broad audience with regular execution schedules and limited user interaction (Kimball *et al.*, 1998:386). Kimball *et al.* (1998:387) describe requirements for standard reporting tools. Some of these include, report development environment, time and event-based scheduling of report execution and flexible report delivery.

After requesting information from the data warehouse, the required data leaves the *presentation server* and ends up on the user's personal computer. Alternatively, the result can be fed into front-end tools from the data warehouse.

Standard reporting tools: These are tools used for creating and displaying reports. Kimball *et al.* (1998:375-376) explain that as transaction systems change into client/server packages, the tasks that should be done by the old report system, are not carried out or are poorly dealt with. This is where the client/server-based standard report tool is used to take advantage of the data warehouse as a primary source. These applications may use multiple data

stores or a report library. Kimball *et al.* (1998:387) describe the requirements and capabilities for standard reporting tools. On the front-end, standard reports also need to provide the same user interface and formatting controls and capabilities as the push button access systems described by Kimball *et al.* (1998:393).

Desktop data access tools: As data moves from the back room to the front room, "it becomes more diffused" and the users can generate queries and reports as often as required (Kimball *et al.*, 1998:375). The results are stored (temporarily) in the data access tool and most of the time the results are transformed into a spreadsheet to be further analyzed (Kimball *et al.*, 1998:375).

Application models: The best example of an application model is data mining (see definition of data staging and data staging area in par. 3.4).

Downstream/operational systems: As the data of the data warehouse increasingly becomes the source of analysis and reporting, "other systems are drawn to it as the data source of choice" (Kimball *et al.*, 1998:378). While most of these systems are transaction oriented, they gain value by including history from the data warehouse. Example: budgeting/forecasting systems. This can help organisations carry out sales transactions with customers. While doing a sales transaction over the phone, the customers' history and credit history is already available. The administrator can then, before selling the product, ascertain whether the customer will be able to pay (i.e. has a good credit record) or not.

3.5.2 Kimball's data warehouse development lifecycle

The business dimensional lifecycle diagram "depicts the sequence of high level tasks required for effective data warehouse design, development, and deployment" (Kimball *et al.*, 1998:33). Throughout the business dimensional lifecycle, Kimball *et al.* (1998:33-39) explains that business requirements are the most important. The process of collecting business requirements differs from Inmon's (1996) view of using a data-driven requirements analysis

methodology. It is important to realise that in the business dimensional lifecycle of developing a data warehouse, project management is an ongoing process that starts with planning and ends with maintenance and growth (see figure 3.4). Each of these guideposts in the diagram will be explained using the requirement-driven methodology of Kimball *et al.* (1998:33-38).

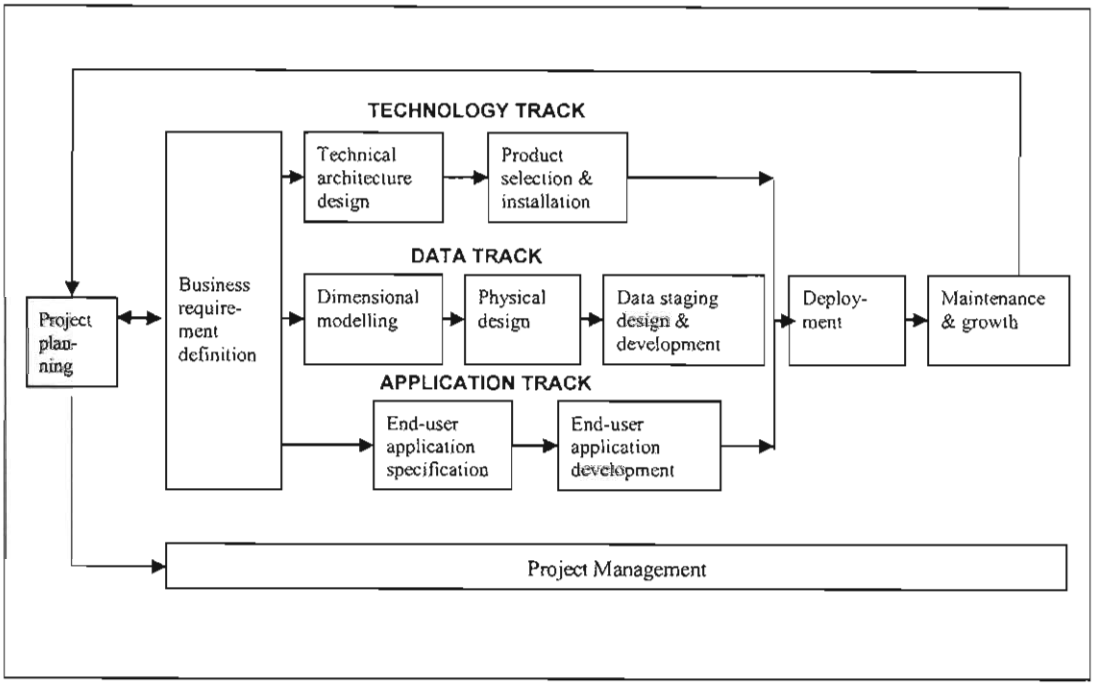


Figure 3.4: The business dimensional lifecycle diagram (adopted from Kimball et al., 1998:33)

Project planning: Project planning is a critical stage in the business dimensional lifecycle because of the costs and scoping process involved in data warehouse development. Planning addresses the project's definition and scope, including business justification and readiness assessment. Thereafter, project planning focuses on staffing requirements (e.g. funding people who can do the work), coupled with task assignments, sequencing and the duration of the tasks to be completed. Planning is the most important factor for ongoing management and it identifies all tasks involved in the business dimensional lifecycle, as well as the people involved.

Business requirements definition: Business requirements establish the foundation for technology, data and end-user applications to create a project of

success. For a data warehouse to be successful, it is important to understand the requirements of business end-users. Designers should understand the key business driving factors in order to determine business requirements and to change them into design considerations.

DATA TRACK

Dimensional modelling: Dimensional modelling is explained in detail in par. 3.4 and par. 3.5.4. Firstly, a matrix is constructed that represents key business processes (data marts) and their dimensionality. A dimensional model is then created analysing data of relevant source systems and business requirements. This model identifies the fact table grain, associated dimensions, attributes, hierarchical drill paths of facts and appropriate table structures with primary/foreign key relationships.

Physical design: The physical data warehouse is designed by defining the physical structures necessary to support the logical data warehouse design. Indexing and partitioning strategies are primarily determined, naming standards are defined and the data warehouse environment is set up.

Data staging design and deployment: Data staging is explained in par.3.4 and 3.5.5. The data staging process (ETL-process) has three major steps: extraction, transformation and load. Two data staging processes should be built, one for the initial population of the data warehouse, and the other for regular incremental loads.

TECHNOLOGY TRACK

Technical architecture design: The overall architecture framework and vision is established. To establish the data warehouse technical architecture design, business requirements, current technical environment, and planned strategic technical directions should be considered simultaneously. (See par 3.5.1).

Product selection and installation: Using the high-level technical architecture

as a framework, components such as the data staging tool or data access tool need to be evaluated and selected. An evaluation process is defined along with specific evaluation factors for each component of the high-level technical architecture. After evaluating and selecting the products, they are installed and tested thoroughly.

APPLICATION TRACK

End user application specification: Application specifications describe the required calculations and user driven parameters. These specifications ensure that the business users and development team understand the applications that should be developed.

End user application development: The users can construct specific reports. The reports are built using an advanced data access tool that provides value to the team. It can also provide a mechanism to easily modify existing report templates.

Deployment: Deployment represents the combination (with extensive planning) of the three tracks, technology, data and end user applications, that can be accessed from the user's desktop. Business users should be educated on all aspects of the combination of the three tracks. User support and communication or feedback strategies should be established before users are granted access to the data warehouse (see par 3.5.6)

Maintenance and growth: Business users should be provided with ongoing support and education. Continue giving attention to the back room (see fig 3.2) to ensure that the procedures and processes are in place for effective ongoing operation of the data warehouse. Using a business dimensional lifecycle, the data warehouse will evolve and grow (a sign of success). A prioritisation process should be established to deal with this business user demand for evolution and growth.

Project management: This step ensures that the business dimensional lifecycle activities remain on track. Project management activities occur throughout the lifecycle, focusing on issue tracking, scope boundaries and monitoring project status. Lastly, project management include the development of a communication plan that addresses information systems and business organisations. To achieve data warehouse goals, ongoing communication is critical to manage expectations. Thus, it would be wise to use methodologies, like ASDMs that focus on communication and people.

The business dimensional lifecycle does not attempt to create an absolute project timeline. Each box in figure 3.4 is merely a guidepost. This lifecycle can be customized and adjusted to address the unique needs of a specific organisation (Kimball *et al.*, 1998:38). According to Kimball *et al.* (1998:39), this lifecycle is "most effective when used to implement projects of manageable yet meaningful scope".

3.5.3 Collecting requirements

According to Kimball *et al.* (1998:95) "business requirements have an impact on every aspect of developing a data warehouse". That is why they follow a requirement-driven methodology.

Kimball *et al.* (1998:96) recommend talking to business users in order to better understand business requirements. The authors (1998:97) further state, "you can't just ask users what data they would like to see in the data warehouse". You should instead be talking about their jobs, their challenges and objectives to try to understand what kind of decisions they must make everyday. Its also important to interview IS (information system) personnel (Kimball *et al.*, 1998:97) to identify and understand their business requirements.

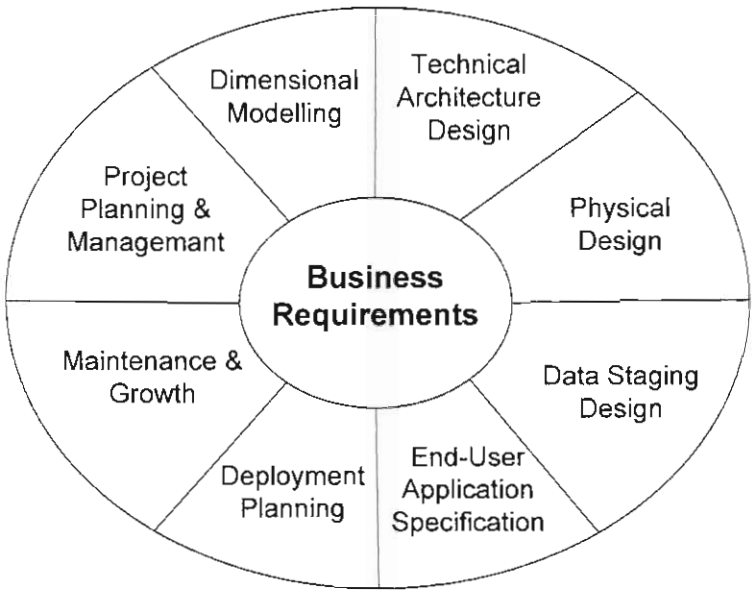


Figure 3.5: Business requirements affect virtually every aspect of the data warehouse project (Kimball et al., 1998:96)

There are two basic techniques for gathering requirements, namely interviews and facilitated sessions. Kimball *et al.* (1998:97-98) suggest that interviews should be conducted in small groups or one-to-one. This leads to participation and detailed data, although it takes a lot of time. Facilitated sessions are shorter but do not give the high level of detailed data that individual or small group interviews do. It also needs a facilitator who prevents extroverts from taking over. Facilitated sessions cause brain storming but it also may result in certain individuals not taking part. This problem can be corrected by interviewing smaller groups.

What should be done first? Kimball *et al.* (1998:98-104) suggest starting by preparing for the interview. This involves identifying an interview team that includes a lead interviewer, who leads the interview; a scribe, who takes notes as the interview progresses; and observers – people interested in the interview. Next, pre-interview research should be conducted, followed by selecting the interviewees in cooperation with IS management sponsors.

Thereafter questionnaires should be developed that "should be structured to align your intended interview flow" (Kimball *et al.*, 1998:104). Then the interviews should be scheduled. Next, the interviews should be sequenced, starting with the business driver and business sponsor. A time and place should also be specified that suit the interviewees. Lastly, the interviewees should be prepared by arranging a kick-off meeting (Kimball *et al.*, 1998:108).

The first thing to remember when conducting the interviews is the roles of every person involved in the interview. The lead interviewer asks the questions, the user answers, the scribe takes notes and the rest should listen.

It is very important to verify that communication took place. One should understand what the interviewee is saying. The terminology should be defined; but vocabulary should not be viewed as unimportant. Furthermore, a peer base should be established where every user is seen as an equal. It is also important to remember that one should maintain the interview's schedule flexibility. Do not let the interview burn out and remember to manage expectations continuously without overselling the data warehouse. (Kimball *et al.*, 1998:111-114).

Start the interview by keeping everything stated above in mind. Business executive interview questions, IS data audit interview questions and business manager analysis interview questions can be asked as the interview progresses. Next, the interview should be properly concluded (Kimball *et al.*, 1998:122-123). Lastly, the interview results should be reviewed in order to determine which requirements are set by the users (Kimball *et al.*, 1998:126-127).

Several common interview obstacles might occur while collecting business requirements. These include (Kimball *et al.*, 1998:124-125):

- *Abused user:* It is a user who is frustrated and who normally says, "We already told IS what we want".

- *Overbooked user:* This happens if all users are too busy to attend interviews, which may result in the project being cancelled.
- *Comatose user:* User responds with single words, and does not perceive the interview as serious.
- *Overzealous user:* A user who arrives determined to be heard (opposite of comatose user).
- *Nonexistent user:* User who thinks he knows best what the IS needs.

After having determined what users want and what is required to satisfy their requirements, the following activities should take place (Kimball *et al.*, 1998:131). Firstly, users have to agree that the collected requirements are accurate. If users are not satisfied with the requirements and want more that can be delivered in a single phase of implementation, the team needs input from the business community. The team needs to reach consensus about the scope before continuing with the project.

3.5.4 Data modelling

The next task, after defining requirements, is data modelling, where all the requirements should be modelled into diagrams. Kimball (1998) uses star-schemas known as dimensional models, and not the traditional ER (entity relationship)-diagrams.

The difference between dimensional modelling and a large ERD is that a single ERD breaks down into multiple fact table diagrams. Kimball *et al.* (1998:146-147) describe three steps to convert an ERD to dimensional model diagrams:

Step 1: Separate the ERD into its discrete business processes and model each business process.

Step 2: Select the many-to-many relationship in the ER-model containing numeric and additive non-key facts and designate them as fact tables.

Step 3: Denormalise the remaining tables into dimension tables with single part

keys that connect directly to the fact tables. Conform dimensions (dimensions shared by dimension models) can be formed where one dimension connects to more than one fact table.

There are many advantages in using star-schemas. Kimball *et al.* (1998:147-149) describe the four main advantages (strengths). Firstly, the dimensional model is a standard, predictable framework. Secondly, a star-schema has the ability to change, which is not the case with an ERD. Thirdly, a dimensional model can accumulate new data elements as well as new design demands and requirements. Lastly, aggregates can be built to ensure early and speedy feedback to users.

Kimball's star-schema can best be illustrated by using an example. Before looking at the example, there is some detail to understand. A dimensional model consists of a fact table and dimension tables. Each dimensional model represents a single business process as well as a data mart. A collection of dimensional models relates to one another and is called a data warehouse.

The fact table is seen as the main table and is placed in the middle of the dimension model. The dimension tables are placed around the fact table, each with its own primary key or surrogate key. The fact table contains all the dimension tables' primary keys or surrogate keys, as foreign keys that are related to every dimension table's primary key or surrogate key. The fact table can also contain one or more facts.

The researcher will explain the development of Kimball's star-schema by using the four steps of designing a fact table given by Kimball *et al.* (1998:194-199):

- **Step 1: Choosing the data mart**

  As stated above, a data mart represents a single business process. In this example, itemized billing will be used. Hypothetically speaking, the example should represent one line on a cell phone (mobile) account.

- **Step 2: Declaring the fact table grain**

  Here the depth of detail is declared (atomic detail). In this case each fact table entry represents a call made by a cell phone (mobile) user of a specific network (only one network, either MTN, Cell C, or Vodacom).

- **Step 3: Choosing the dimensions**

  As explained, the dimensions are the tables around the fact table that can be chosen using the next five questions:

  - Who: Who is involved? Account holder, network called.
  - Why: Why will it (call) be done? Service, package, cost structure.
  - Where: Where will it be done? Original location, destination location.
  - When: When will it? Date.
  - What: What will be done? Promotion.

  According to Kimball *et al.* (1998), the designer of the dimensional model examines all the data sources available and preferentially attaches the single-valued descriptors as dimensions. Each dimension table has its own granularity that cannot be lower than the total granularity of the fact table.

- **Step 4: Choosing the facts**

  As stated in the definitions, facts are things one does not know until it happens. In this example, the following facts can be declared:

  - Number called
  - Start time (of call)
  - End time
  - Duration
  - Service cost (sms, mms)
  - VAT

  Note: This example uses surrogate keys.

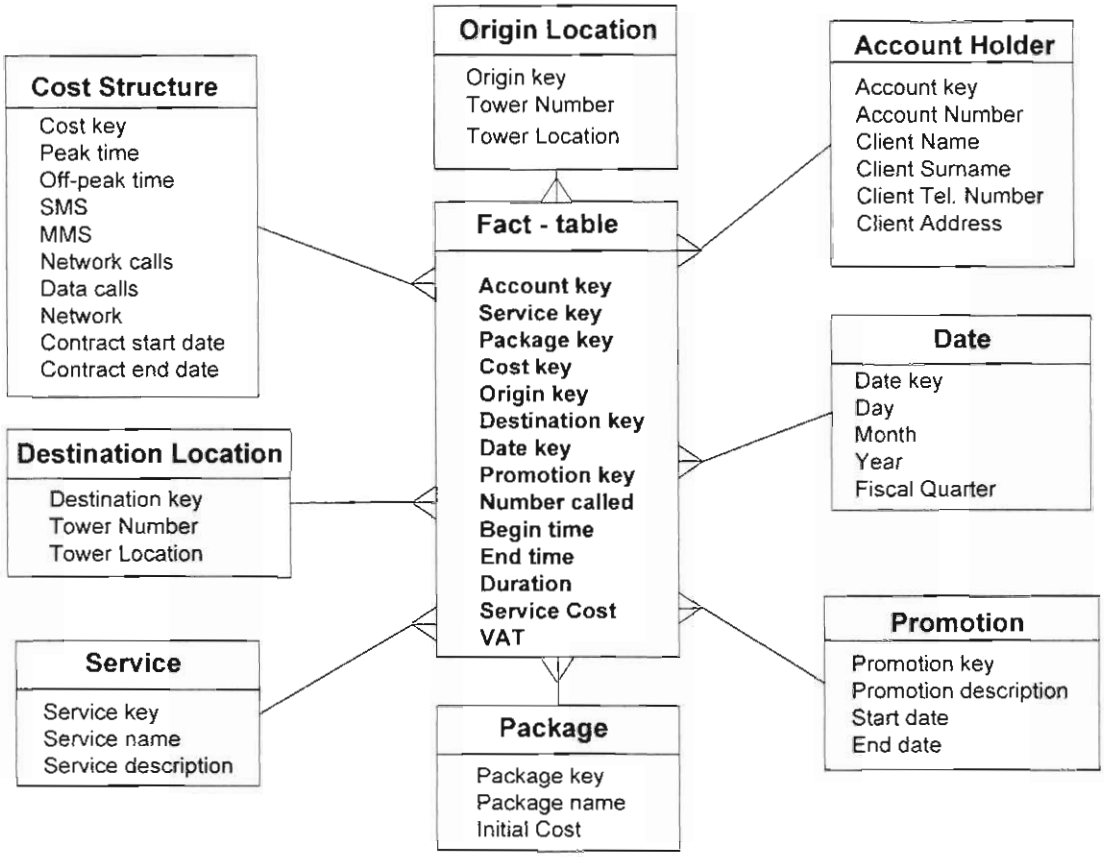


Figure 3.6: A star-schema for itemized billing cell phone (mobile) example

3.5.5 Data staging

Data staging has been defined in par. 3.4. and data staging is explained in par. 3.5.1 and par. 3.5.2.

Data staging is also known as the ETL-process (extract, transform, load) as stated earlier. During the data staging process, data is extracted from the data source system, transformed according to data warehouse standards and cleansed before loading the transformed data into the data warehouse. In short, data staging is the shifting of data from the operational database to the data warehouse.

Kimball *et al.* (1998:23) explain the ETL-process as follows. During extraction,

the data that is required is copied from the source system to the data staging area for further work. After extraction, transformation is done where the data is cleaned, data sources are combined, surrogate keys are created for each dimension and aggregates are built to enhance query performance. Lastly, the fact and dimension tables are presented to the bulk loader that loads it record by record into the data warehouse. A detailed presentation of the ETL-process is given in fig. 3.7.

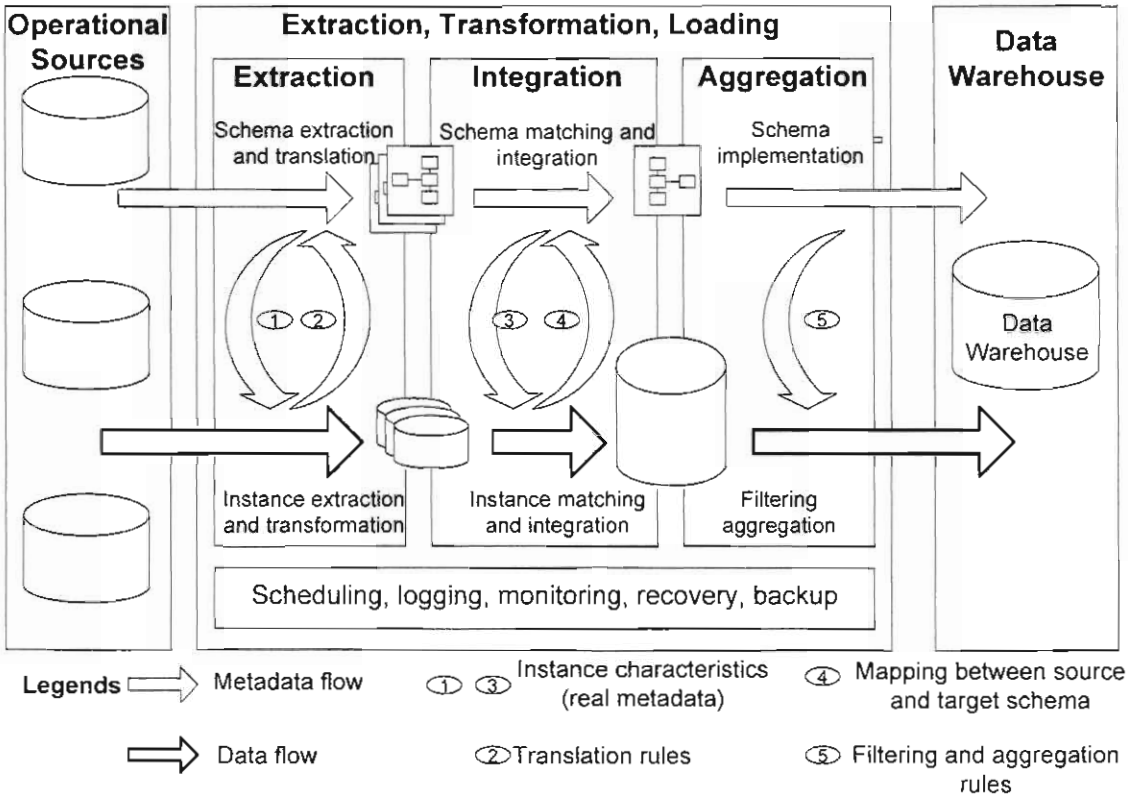


Figure 3.7: Steps of building a data warehouse: the ETL-process (Rahm & Do, 2000:1).

Data staging will be explained by following the 10 steps of the ETL-process, explained by Kimball *et al.* (1998:612-650). Steps 1, 2 and 3 cover planning the process effectively. It includes the decision on a data staging tool (Kimball *et al.*, 1998:166). During the dimension table staging phase, steps 4, 5 and 6 are covered. This entails the building of the data staging application by choosing and using the simplest dimensions (Kimball *et al.*, 1998:167). Lastly, the fact

table loads and warehouse operations phase covers steps 7-10, where incremental loads are suggested for fact- and dimension tables that are too large for a single load process (Kimball *et al.*, 1998:680).

- **Step 1: High-level plan**

  Start the design process by putting the pieces you know of in a simple schematic format containing only sources and targets. Keep the design on one page, indicating were the data is coming from, and including the challenges and requirements you already know about. Use placeholders for unknowns.

- **Step 2: Data staging tools**

  It is very important to choose a data staging tool and to use it early in the project. New releases of data staging tools are showing significant functionality and usability. The team should make a decision whether to use hard coding for extraction or approved data staging tools.

- **Step 3: Detailed plan**

  "Start planning which tables to work on, in which order, and for sequencing the transformations within each data set" (Kimball *et al.*, 1998:615). Graphically diagram the complex restructurings into a set of fact tables onto a single page. Structure the diagrams around the source tables instead of the target tables. The data staging area should also be organized, so that the raw data that has been loaded can be cleaned, combined, archived and exported to the presentation server.

- **Step 4: Populate a single dimension table**

  The main reason starting with a static dimension table is that it is the easiest table to populate. In order to populate a static dimension table, the primary source of data (lookup table) should be extracted to the staging area, after which the data is cleansed during the transformation process. Lastly, the cleansed data is loaded into the target tables using the bulk loader.

- **Step 5: Implement dimension change logic**

  Every data warehouse key should be a surrogate key in the data warehouse DBA (i.e. in every data mart or star-schema) in order to respond to changing descriptions and abnormal conditions in the raw data. If the join key between dimension tables and fact tables are direct derivations of a production key, change will not be possible. The implementation of dimension change logic can be done by doing dimension table extracts, processing slowly changing dimensions and by transforming and loading these slowly changing dimensions.

- **Step 6: Populate remaining dimensions**

  After populating a simple dimension table (step 4) and implementing dimension change logic (step 5), the rest of the dimensions should be populated. At this point populating the remaining dimensions will be easy, unless there are major data quality issues.

- **Step 7: Historical load of atomic-level facts**

  This is done by historical fact table extracts, where records should be identified that fall within the basic parameter of the extract, and where the records are of value for the data warehouse. This is also done by fact table processing, where surrogate keys replace the production IDs in the incoming fact table.

- **Step 8: Incremental fact table staging**

  The problem with most fact tables is that they become too big. The data cannot be loaded into the fact table all at once. A common technique that can be used is to only load the most recent data or only the data that has changed. This means that incremental loading of data into fact tables is used.

- **Step 9: Aggregate table and MOLAP loads**

  The fact table can be too large to be loaded all at once, so it can be a

problem building aggregate tables from a query on the fact table. This problem can be solved by doing aggregate table incremental loads. The aggregates are merely results of big queries mostly including the latest data.

- **Step 10: Warehouse operation and automation**

  The ultimate warehouse operation would run the regular load process completely unattended. Although this is difficult to attain, it is possible. Kimball *et al.* (1998:650) describe different operational functions and a few approaches to implement these functions. The ongoing operations functions of the data warehouse can be automated (semi-automated) by using a good data staging tool.

In order to get quality data in the data warehouse, the data gathering process should be well designed and the people (resources), providing the data should deliver quality information (Kimball *et al.*, 1998:653). Cleaning data uses two processes, namely entering clean data and correcting the problems once the data has been entered. No data warehouse has perfect data, but Kimball *et al.* (1998:653) explain quality data as accurate, complete, consistent, unique, timely and truthful.

The data quality contained in the data warehouse after the loading process can be assured by using techniques like cross-footing, manual examination and process validation (Kimball *et al.*, 1998:658-659). Cross-footing is a query that is run against the source system. The results are then compared with the results of the same query against the load set. Manual examination involves data checks to try to find errors, while process validation may show that the data warehouse might be slightly different from the source system. It will, however, be close enough.

3.5.6 Data access and deployment

This phase contains two parts, data access and deployment. A way should be

found for users to access data easily and to get results that are of value to them. Access applications are created for business users in a manner that enables them to locate the necessary data as speedily and easily as possible in order to analyse the data so it can be of value to the organisation. OLAP (see par.3.) is used to create applications to do queries on data in the data warehouse. According to Goede (2005:153) "tools for end user access focus on trend analysis and ad hoc queries".

Deployment takes place when the data warehouse is completed and the end users use it and find it to be of value. The users should not be negative towards the data warehouse; otherwise, they will not use it. They should understand the power and effectiveness of a data warehouse in order to gain value. To solve the problem of negativity, towards the data warehouse it is important that users are involved from the beginning so that they adapt ownership that will ensure motivation and successful implementation of the data warehouse.

It is important to educate the end-users (Kimball *et al.*, 1998:693) in order for them to gain maximum value out of the data warehouse. It is no use to design a data warehouse with a very sophisticated architecture, including large amounts of clean data, and the user does not even know how to perform a query.

It is also important to implement a business end user strategy to support users after deployment (Kimball *et al.*, 1998:699). To deploy the data warehouse, a framework should be built that encapsulates the pieces involved during deployment. The data warehouse should go through an internal alpha test period (internally test all the components), followed by a beta test period (giving access to a limited number of business users) before the data warehouse is generally available (Kimball *et al.*, 1998:705).

After implementation, it will be interesting to study the use and effectiveness of the implemented data warehouse. This can be done by submitting questionnaires electronically to all users.

Maintaining and managing the data warehouse after successful deployment, as well as preparing for the growth and evolution of the data warehouse, is an ongoing process. After the project priorities have been identified, the lifecycle is run from the start; building upon what has already been established, and focusing on new requirements and recommendations. (Kimball *et al.*, 1998:37). If the data warehouse grows and evolves, the design can be declared successful. "Success breeds success" (Kimball *et al.*, 1998:733).

3.6 Inmon's approach towards data warehouse development

Inmon's approach towards the development of a data warehouse, differs greatly in both view and design from Kimball's approach. During the discussion of Inmon's approach, the researcher will use Kimball's approach towards certain components in data warehousing as a reference to explain Inmon's data warehouse components.

3.6.1 Hub-and-spoke architecture

Inmon follows a hub-and-spoke architecture, while Kimball uses the high-level technical architecture. Inmon's data warehouse can be seen as the central data warehouse or hub. Using Inmon's definition for a data warehouse (see par 3.3), TDWI (2004:27) state that the definition of a hub-and-spoke architecture "serves as a single source hub of integrated data upon which all downstream data stores are dependant".

The data marts are populated from a single integrated and consistent source. Inmon, just like Kimball, uses an ETL-process were the data from the source is transformed using a specific standard of transformation and loading it into the central data warehouse or hub.

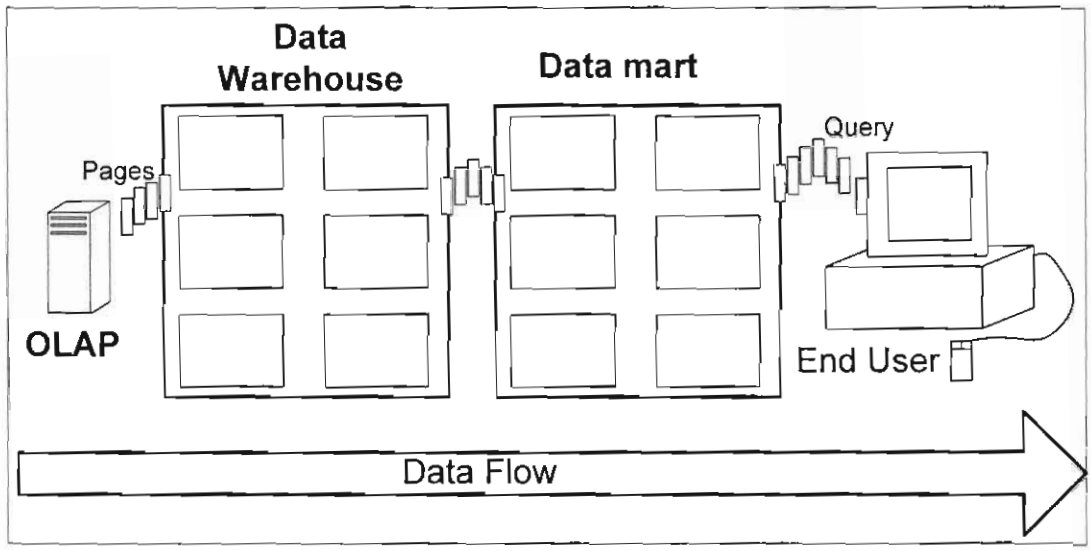


Figure 3.8: Inmon's architecture approach (adopted from Mailvaganam, 2004).

In figure 3.8, the data of Inmon's data warehouse is integrated and a data mart is seen as an interface between the end-user and data warehouse and not as a "logical sub-set of the complete data warehouse", as defined by Kimball *et al.* (1998:19). Inmon (1996) follows a bottom–up approach, i.e. the data warehouse is firstly built and then requirements will evolve and become known.

TDWI (2004:29) provides advantages and disadvantages for using a hub-and-spoke architecture:

Advantages	**Disadvantages**
• Produces a flexible enterprise architecture • Retains detail data in relational form • Eliminates redundant extracts from operational data sources • Integration is consistent and enforced across data marts	• Requires considerable front end analysis - long start-up time • Warehouse grows large quickly - high start-up costs and maintenance • Design to delivery time is too long

Table 3.1: Advantages and disadvantages of hub-and-spoke architecture (TDWI, 2004:29)

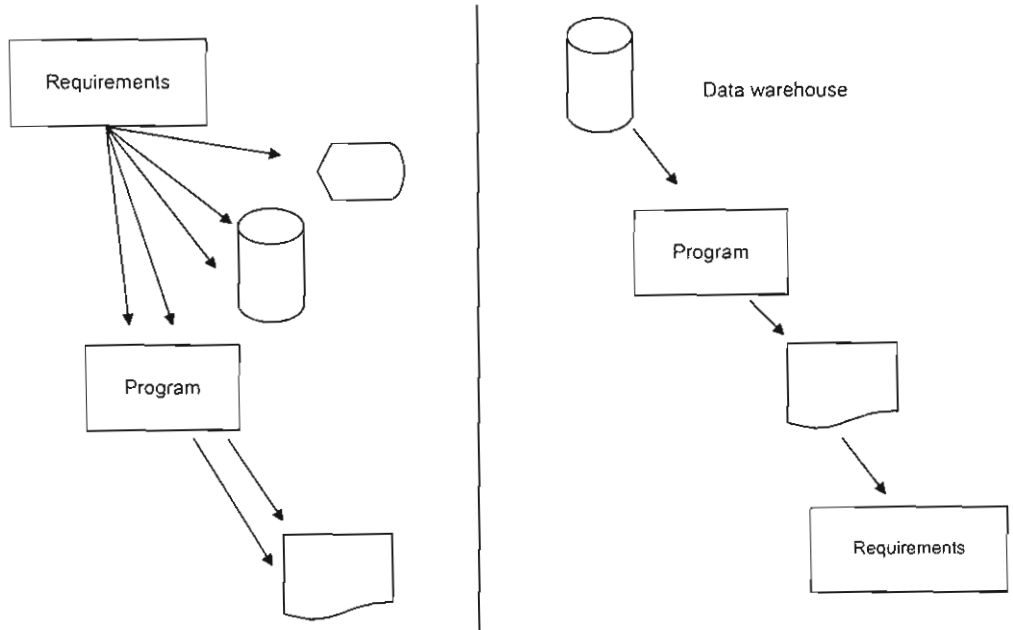
Inmon (1996:20) defines four levels in the architected environment, namely operational, atomic or data warehouse, departmental and individual. The operational level holds primitive data that serves the high-performance transaction processing community. The data warehouse level holds primitive data that is not updated, while the department level contains exclusive derived data. The individual level is where heuristic analysis takes place. At first glance, it may seem as though the architected environment contains redundant data, but the architected environment in fact has integrated data.

3.6.2 Inmon's data warehouse development lifecycle

Inmon's data warehouse development lifecycle follows a data-driven methodology, while Kimball's lifecycle is requirement-driven. According to Inmon (1996:44), a central data store for one subject area that is populated with operational systems should be built,. The demand for an integrated data store for another subject area will grow as the analytical ability of the new data warehouse is discovered. This process will repeat itself until a complete data warehouse has been developed (Goede, 2005:142).

Figure 3.9 presents the classical system development lifecycle (SDLC) as well as the CLDS (data warehouse SDLC). The SDLC is requirement-driven, just like Kimball's approach where requirements should be understood before the stages of design and development are implemented in order to build the system (Inmon, 1996:25).

The CLDS (reverse of SDLC) is data-driven, which means that data is the most important. CLDS starts with integrated and tested data, whereafter programs are written that uses the data. The results of the programs are analysed to identify and understand the requirements (Inmon, 1996:25). Inmon uses the CLDS data-driven approach. He believes requirements are only understood if the data warehouse has already been developed.



Classical systems development lifecycle (SDLC)

- Requirements gathering
- Analysis
- Design
- Programming
- Testing
- Integration
- Implementation

Data warehouse SDLC

- Implement data warehouse
- Integrate data
- Test for bias
- Program against data
- Design DSS system
- Analyze results
- Understand requirements

Figure 3.9: The system development lifecycle for the data warehouse environment (Inmon, 1996:24)

3.6.3 Collecting requirements

According to Inmon's approach, a data warehouse should be developed using existing data to satisfy decision makers before requirements are identified and satisfied. As seen in figure 3.9, Kimball's view of collecting business requirements is very important. In contrast, Inmon (1996:144) states that, "requirements for the data warehouse cannot be known a priori".

Using Inmon's CLDS, requirements are identified last and not first as in the case of the traditional SDLC. The time to complete data warehouse development and user requirements depend on the size of the project. The

larger the data warehouse project, the longer it will take to meet requirements.

3.6.4 Data modelling

Inmon (1996:85) proposes that an ERD data model be used, instead of Kimball's star-schema for the development of a data warehouse. A combination of individual ERDs, where each reflects the different views of people within the organisation, makes up the corporate ERD. Inmon (1996:82) states that in order to construct a data model, the corporate model should be used as starting point. The feedback loop explained in par. 3.6.6 can be used to identify DSS-analyst requirements.

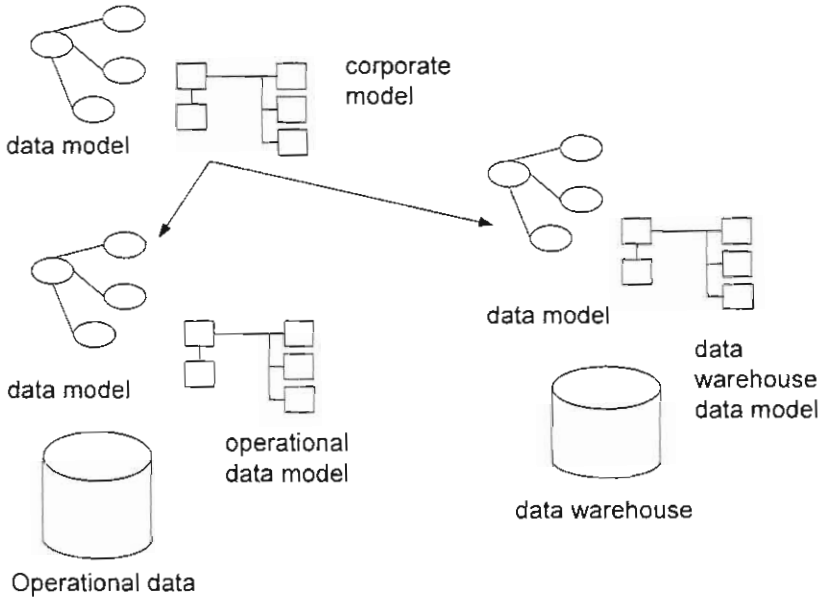


Figure 3.10: Operational data model vs. data warehouse data model (Inmon, 1996:83)

In order to construct a data warehouse data model, the pure operational data should be removed followed by the enhancement of the key structures by adding an element of time to each key. Derived data that is publicly used, is added to the corporate data model only once it has been calculated. Lastly, the operational environment's relationships are turned into "artefacts" in the data warehouse.

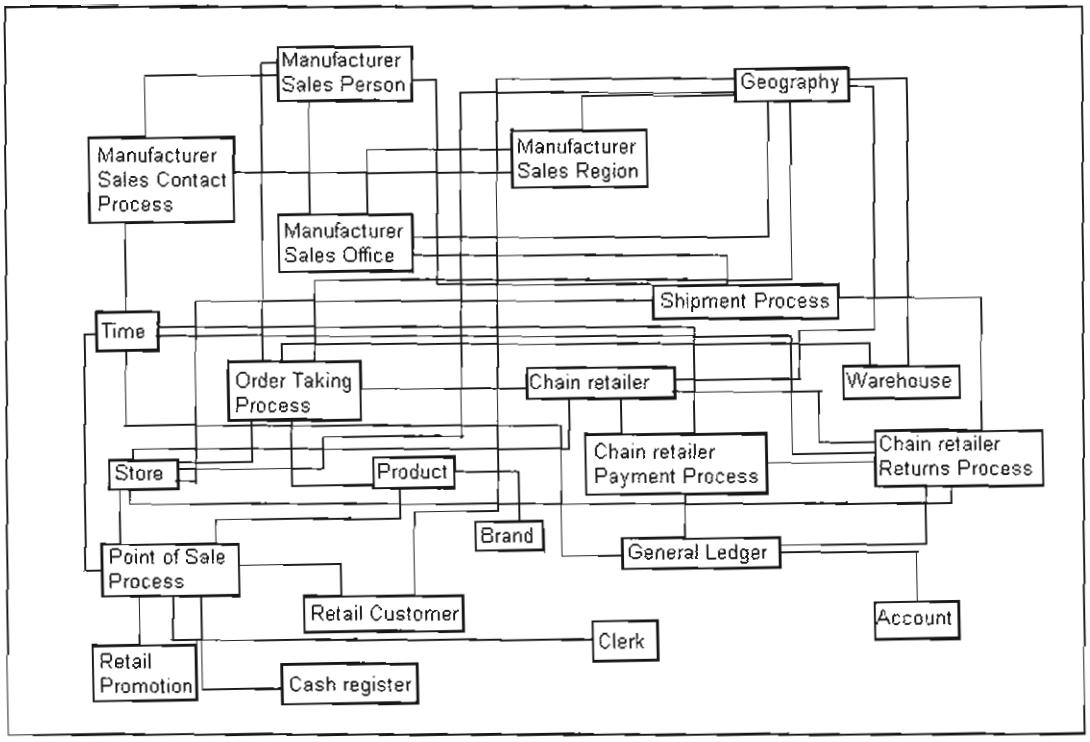


Figure 3.11: An entity relationship model of an enterprise that manufactures goods (Kimball et al., 1998:143).

Inmon (1996:139) also describes star-schemas, but calls them star-joints. Inmon emphasises that star-joints and ERDs will lead to an optimal data warehouse design. According to Goede (2005:147), Inmon (1996) does not offer sufficient explanation on how this is achieved.

According to Inmon (1996:85), there are three levels of data modelling:

- **High-level modelling (ERD: entity relationship diagram)**
- **Mid-level modelling (DIS: data item set)**
- **Low-level modelling (physical model)**

High-level modelling can be done by surrounding every entity with an oval and representing relationships between entities with arrows, where the number of arrowheads indicate the cardinality of the direct relationship. As stated earlier,

the corporate ERD of the data warehouse is a composite of many individual ERDs that reflect the different views of people across the organisation.

The mid-level model is created after the high-level model. For each entity in the high-level model, a mid-level model is created. The mid-level data model for one entity or major subject area is explained, "then the mid-level model is fleshed out while other parts of the model remain static" (Inmon, 1996:88). A mid-level model is created for every entity or major subject area. Four basic constraints should be remembered in the creation of a mid-level model:

- *A primary grouping of data:* Contains attributes and keys, where the attributes are held that only exist once for every entity.
- *A secondary grouping of data:* Holds attributes that can exist more than once for each entity.
- *A connector:* The connector relates data between groupings of data. To indicate a connector, the foreign key is underlined.
- *"Type of" data:* Indicated by a line leading to the right, where the group of data to the right is sub-type data and the group of data to the left is super-type data, indicates the "type of" data.

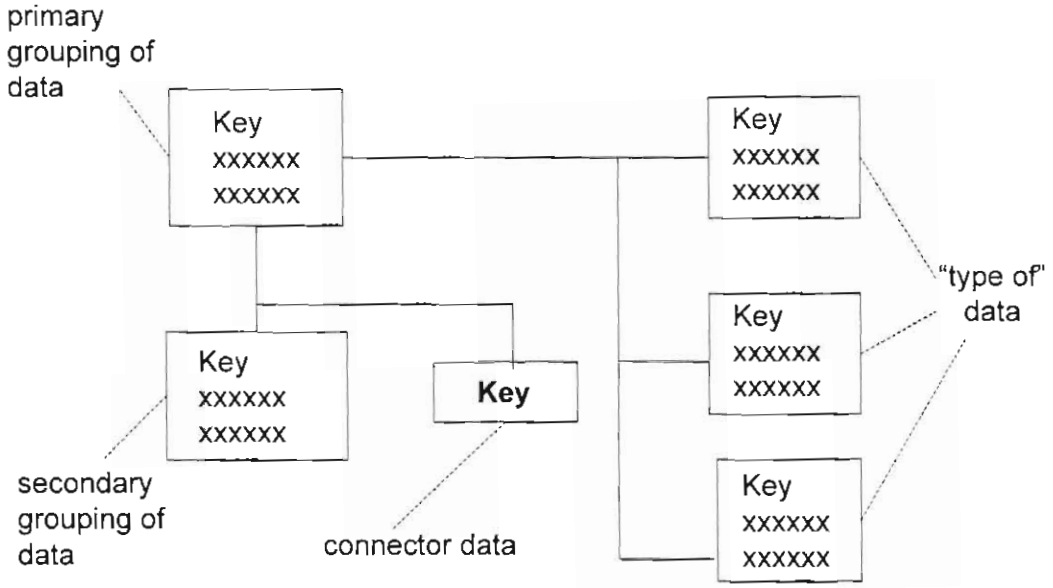


Figure 3.12: The four constraints of the mid-level data model (Inmon, 1996:89)

The low-level data model is created from the mid-level data model by expanding the latter to include keys and physical characteristics of the mid-level model. In the physical (low-level) model the model looks like a series of tables, called relation tables. One very important design step remains, namely "factoring in the performance characteristics" (Inmon, 1996:93). This means deciding on the granularity and partitioning of the data being used. After this is done, other physical design activities can be used in the design.

The problem with the data model is that is appears to make all the entities peer with each other. To get data to provide a three-dimensional perspective, a star-join can be used.

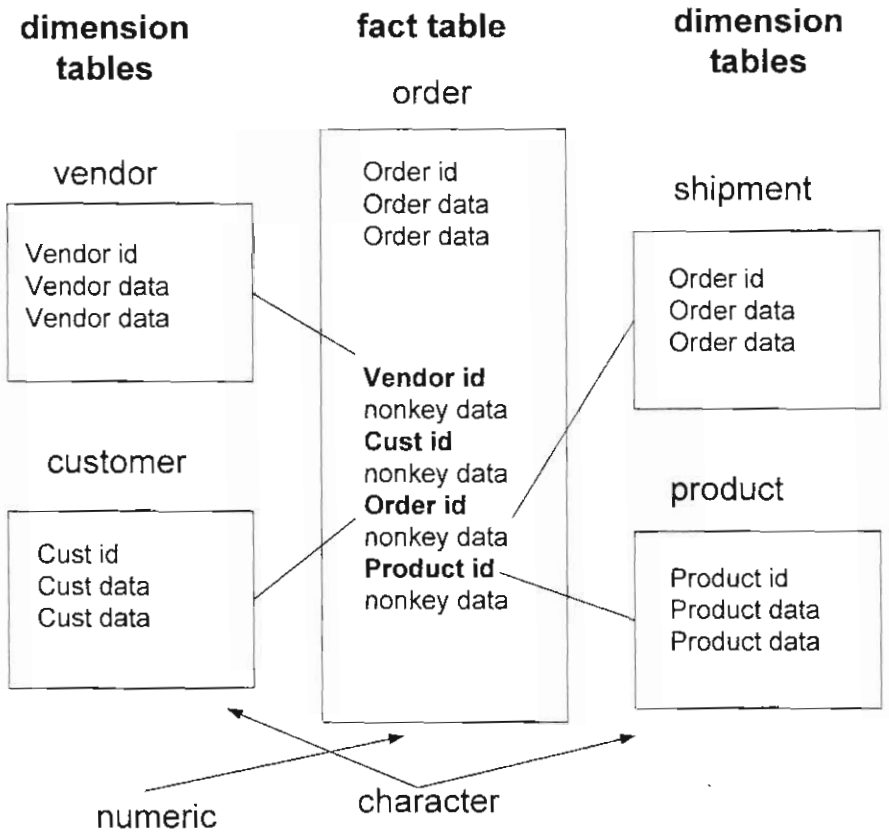


Figure 3.13: Star-join with fact table and corresponding dimension tables (Inmon, 1996:142)

A star-join is a "design structure that is required to manage large amounts of

data residing in an entity in the data warehouse" (Inmon, 1996:140). Figure 3.13 is an example of a star-join that resembles Kimball's (1998) star-schema. The *order* table in the middle of the star-join is called the fact table. This entity is heavily populated. The surrounding tables, *vendor, customer, shipment* and *product,* are called *dimension* tables.

The fact table (*order*) contains data unique to the fact table itself, as well as unique identifying data for the fact table. The fact table also contains foreign keys that references to the surrounding dimension tables. The star-join can also contain non-foreign key information, but this information should be used frequently with the fact table.

During the creation of a star-join, the textual data is often separated from numeric data (this is done in Kimball's star-schema by using surrogate keys). Normally textual data ends up in the dimension tables and numeric data is seen in the fact table (see figure 3.13).

The star-join is used within the DSS data environment. If star joins were used outside the DSS where data relationships are managed and updated, it will be a very cumbersome structure (Inmon, 1996:142).

The benefit of using star-joins is to "streamline data for DSS processing" (Inmon, 1996:142). By creating selective redundancy and by pre-joining data, the data is simplified and streamlined for access and analysis, "which is exactly what is required for the data warehouse" (Inmon, 1996:142).

3.6.5 Data staging

This data staging heading can also be replaced with data cleaning or data cleansing. Data cleansing is used to see if business, user and analyst requirements are met. Inmon (1996) also uses the ETL-process were data is extracted from the source, transformed according to organisation's standards

and loaded into the central data warehouse (TDWI, 2004:29).

Because the data warehouse of Inmon is integrated, transformation should be mapped from the different source fields to the data warehouse fields. Not only is integration difficult when transforming an existing systems environment to the data warehouse environment, but the efficiency of accessing existing system data is also hampered (Inmon, 1998:76). There are three types of loads from the operational environment to the data warehouse environment (Inmon, 1996:76):

- The loading of archival data.
- The loading of data contained in the operational environment at that point in time.
- The loading of ongoing changes to the data warehouse environment from the changes (updates) that have occurred in the operational environment since the last refreshing of the data warehouse.

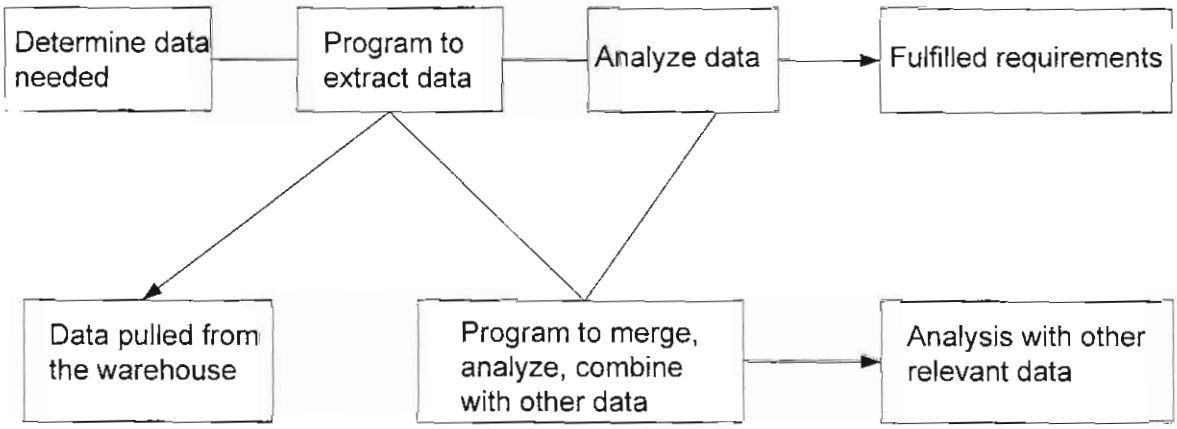


Figure 3.13: Activities for cleaning data (Inmon, 1996:357)

Inmon's data cleansing activities begin by determining the required data and ending with institutionalisation. The data required should firstly be determined. "Data in the data warehouse is selected for potential usage in the satisfaction of reporting requirements" (Inmon, 1996:347).

The first few times this activity is executed, only part of the required data will be retrieved from the data warehouse. It is important to remember that Inmon's approach is all about data (data analysis). This means that data is seen as most important. The data that is selected can be used for further analysis.

Furthermore, a program should be written to extract data from the data warehouse after selecting the correct data. This is done by writing a program to "access and strip" the data (Inmon, 1996:347).The program should be modifiable when necessary, because most code will be run and modified numerous times. The program is used to pull data from the data warehouse for DSS analysis.

The data should then be prepared for analysis by combining, merging and analysing the data after selection has taken place. This means combining, editing and refining data for analysis (Inmon, 1996:348).

Subsequently, the data should be analyzed. The question here is: "Do the results obtained meet the needs of the analyst?" (Inmon, 1996:348). If the results satisfy the needs of the analyst, the preparation for the final report can commence. If the result does not satisfy the needs of the analyst, it causes another iteration to take place.

The final report is then prepared (the questions are answered) and produced, which contains the results of many iterations of processing, as well as the conclusion.

Finally, a decision should be made to determine whether the final report should be institutionalised. If the need arises to run the report repeatedly, it will be a good idea to submit the report as a set of requirements and to rebuild it as a regularly occurring operation (Inmon, 1996:348).

3.6.6 Data access and deployment

According to Inmon (1996:128), there are two kinds of data warehouse data access: direct access and indirect access.

Direct access of data warehouse data

Direct access takes place where a request is made within the operational environment for data that is in the warehouse. The request made by the user or manager is transferred to the data warehouse environment. The requested data is then located and sent back to the operational environment. A scenario can occur where a manager needs to trace data from the data warehouse back to its operational source. This is called the "drill-down" process (Inmon, 1996:186).

There are some limitations to the scenario of direct access (Inmon, 1996:128):

- The request should be casual in terms of response time.
- The request for data needs to be for a minimal amount of data.
- The technology managing the data warehouse needs to be compatible with the technology managing the operational environment in terms of capacity, protocol, etc.
- The formatting of data after it is retrieved from the data warehouse in preparation for transport to the operational environment, should be non-existent (or minimal).

These conditions do not include data that will be directly transferred from the data warehouse to the operational environment.

Indirect access of data warehouse data

Direct access of data warehouse data is one of the most effective uses of warehouse data by the operational environment (Inmon, 1996:129). Inmon (1996:129-138) describes three examples of indirect access of data warehouse data, but in this study only one of these examples will be discussed, namely the air commission calculation system.

In this example (Inmon, 1996:129-131) a travel agent contacts the airline reservation clerk on behalf of a customer. The main concern is the commission paid by the airline. If the commission rate is high, the airline could secure business, but lose money in the process. If the airline pays commission under average, it could lose business. It is therefore very important to carefully calculate the commission rate.

The business process between the client and airline should be as short as possible, because the airline will lose business if response time is poor. Interaction between the travel agent and airline clerk should consequently be as fast as possible. The optimal commission for this example can be computed by using two factors; existing bookings and the load history of the flight, that "yields a perspective of how the flight has been booked in the past" (Inmon, 1996:130).

The calculation of the appropriate commission is done offline to improve response time. Offline calculation is done periodically and a small, simple table to access flight status is created. When the airline clerk interacts with the travel agent, a quick decision can be made by just looking at the current booking and flight status table. The data warehouse is deployed after alpha and beta testing has been completed and the data warehouse is populated from the existing system.

Inmon (1996:283) describes a *feedback loop* (see fig. 3.15) after the data warehouse has been deployed (data warehouse populated from existing system). The DSS analyst uses the data warehouse and analyses new requirements that has been provided to the data architect. The data architect makes appropriate adjustments to satisfy the requirements.

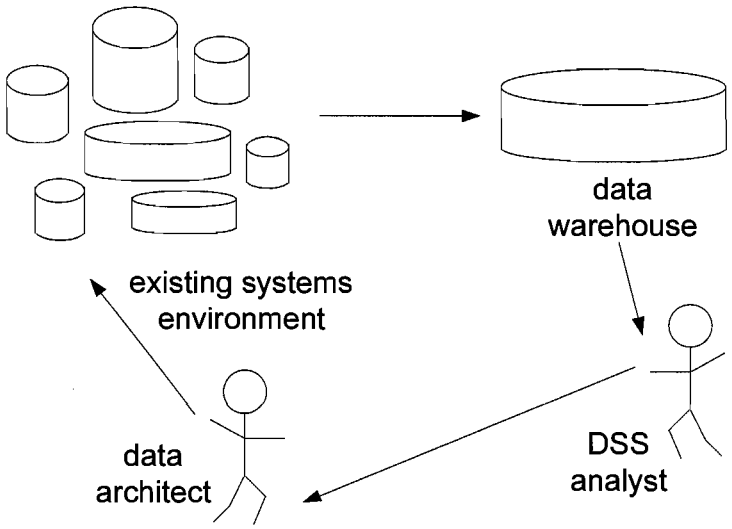


Figure 3.15: Feedback loop between data architect and DSS analyst (Inmon, 1996:283).

3.7 Kimball versus Inmon

Kimball and Inmon have different views towards their respective architectures, data warehousing lifecycles and it's main stages; requirement collection, data modelling, data staging and data access and deployment, as described in par 3.5 and 3.6.

In par.3.3, Kimball's definition of a data warehouse differs from Inmon's definition. Kimball describes a data warehouse as a collection of data marts, where every data mart represents a business process or dimensional mode, while Inmon describes a data mart as an interface between the end-user and the data warehouse.

Inmon's (1996) approach towards the development of a data warehouse follows a data-driven methodology, were data is seen as most important. Inmon believes that the data warehouse should be built using the existing operational system. After successful population of data into the data warehouse, reports are created. After report generation, requirements will be known and implemented using the data architect and DSS analyst feedback loop. Kimball

(1998), on the other hand, follows a requirement-driven methodology where requirements should first be collected from trusted sources to develop a data warehouse that will be of value to business users. After successful deployment, maintenance is done to ensure that the data warehouse is up to date.

It is clear that Kimball works with a top-down approach (starting with requirements), while Inmon follows a bottom-up approach (ending with requirements). Other differences between the approaches include that the conformed dimensions of Kimball are denormalised, while Inmon prefers to use a highly normalised central database model. The dimensions of Kimball used in the data marts are the dimension tables themselves, and not copies of conformed dimensions, while Inmon's data marts store a second copy of the data from the centralised data warehouse tables that take up extra space. Kimball *et al.* (1998:153,347) refer to the data warehouse bus as a set of conformed dimensions.

In examining the data warehousing architecture, it is clear that both Kimball *et al.* (1998) and Inmon (1996) obtain source data from legacy batch and online operational systems and specialised operational data stores (ODS). Inmon and Kimball mainly differ in the arrangement of data in the data warehouse itself. Inmon uses the hub-and-spoke architecture, while Kimball uses the bus architecture. Inmon's approach uses an atomic level, third normal form (3NF) relational format in which to store extracted and transformed data, while Kimball's approach uses a multi-dimensional style containing dimensions and facts (Lawyer & Chowdhury, 2004:2).

Lawyer and Chowdhury (2004:2) examined the differences between the Kimball and Inmon approaches. Kimball's approach has data pre-arranged by certain dimensions according to the desired output, while Inmon's approach is considered application neutral. Kimball's "enterprise" data warehouse is the sum of all individual multi-dimensional data structures; while Inmon's "enterprise" data warehouse has data that covers most data subjects from the

organisation. Data is summarised by higher-level dimensions in Kimball's approach, while in Inmon's approach data is kept at the lowest level of detail, meaning that each transaction would be stored in 3NF (normal form). In Kimball's approach, data is arranged in an application or data-view-specific manner, while in Inmon's approach data is arranged according to the rules of normalisation and remain application and data-view-independent. Legacy system data is neither integrated nor standardised. Sourcing the legacy system would require summarising and arranging facts by their dimensions and standardisation in Kimball's approach, while Inmon uses only standardisation when outsourcing takes place. Authoring SQL to access data arranged in a multi-dimensional database would be a very difficult task using Kimball's approach. Inmon's approach, on the other hand, will simplify the process of accessing the multi-dimensional structure and performing drilling navigation. Lawyer and Chowdhury (2004:2) describes another advantage of Inmon's approach as "the ability to create dependent data marts from the atomic data warehouse for those situations where a repetitive reporting requirement or application-specified need exists".

Inmon (1996:87) prefers the use of ERD data models because it reflects the different views of people across the organisation, as well as a star-join, for it will lead to a successful data warehouse. Kimball prefers the use of a star-schema, where every star-schema is a dimensional model or data mart, presenting a single business process that contains a centralised fact table and surrounding dimension tables. The data warehouse is thus built out of a collection of interacting dimensional models.

TDWI (2004:27) identifies some differences between Inmon's data warehouse and Kimball's data warehouse by looking at key aspects in the development of a data warehouse:

	Inmon warehouse	Kimball warehouse
Intake	Fills the intake role, but may be downstream from staging.	Fills the intake roll – downstream from "backroom" transient staging
Integration	Primary integration data store with data at the atomic level	Integration through standards and conformity of data marts
Distribution	Designed and optimised for distribution to data marts	Distribution is insignificant because data marts are a sub-set of the data warehouse
Access	May provide limited data access to some "power" users	Specifically designed for business access and analysis
Delivery	Not designed or intended for delivery	Supports delivery of information to the business

Table 3.2: Kimball vs. Inmon (TDWI, 2004:27)

In practice, organisations usually pick one approach depending on assumed preference, budget, consultant or vendor recommendation, existing technologies or advantages of the approach.

In this chapter, BI was firstly discussed using the BI framework and explaining data warehousing as a tool of BI. Secondly, a data warehouse was defined, with Kimball *et al.* (1998:27) stating that a data warehouse consists of a collection of data marts, and Inmon stating that a data warehouse is subject oriented, integrated, time variant and non-volatile. After defining a data warehouse, the basic terms associated with data warehousing were defined in par. 3.4.

Subsequently the two different approaches of Kimball *et al.* (1998) and Inmon

(1996) were described by examining six focus areas, namely architecture (par. 3.5.1, 3.6.1), data warehouse development lifecycle (par.3.5.2, 3.6.2), collecting of requirements (par. 3.5.3, 3.6.3), data modelling (par. 3.5.4, 3.6.4), data staging (par.3.5.5, 3.6.5) and data access and deployment (par. 3.5.6, 3.6.6). Kimball's approach is explained as a requirements-driven approach, where collecting requirements are viewed as the most important, and completed first. Inmon's approach is explained as a data-driven approach, where data is seen as most important and requirements are identified after the data warehouse has been developed.

In conclusion, Inmon and Kimball's approaches were discussed by examining their differences and contrasts in both view and design. One of these approaches will be used to develop a data warehouse using a single or selected combination of ASDMs.

In the next chapter, the researcher aims to create a framework in which the seven ASDMs, discussed in chapter 2, will be evaluated by investigating their suitability to each phase of data warehouse development for Inmon and Kimball's approaches, if possible.

CHAPTER 4

THEORETICAL DEDUCTIONS: SUITABILITY OF ASDMs FOR DATA WAREHOUSE DEVELOPMENT

4.1 Introduction

In this chapter, the suitability of the use of ASDMs in data warehousing will be investigated form a theoretical point of view. The general findings associated with the characteristics of all ASDMs in data warehouse development will be explained (par 4.2.1), including the applicability of the nine core values of all agile processes, as explained in par. 2.3. Next the explanation of each ASDM's suitability towards data warehouse development will follow. The findings will be paraphrased under each ASDM for Kimball *et al.* (1998) and Inmon's (1996) approaches.

The words increment and iterative will be used frequently in this chapter. An increment in a data warehousing project can be seen as a data mart or a sub-data mart (sub-division of a data mart), while an iteration can be viewed as a repeatable process in a specific increment. In addition, when referring to a data mart, it includes everything associated with the data mart: the diagram of the requirements into a star-schema (data mart); the design and data staging of the star-schema; the graphical user interface (GUI) associated with the data mart, and the reports that should be generated according to user specifications. Therefore a sub-data mart can be any of the above mentioned sub-divisions of the data mart.

4.2 Kimball's approach

In this section, each of the seven ASDMs discussed in chapter 2 will be investigated from a data warehousing point of view. Their suitability for the use in data warehouse projects will be described in terms of suitable and unsuitable

characteristics, The discussion of each ASDM will be done according to the data warehouse lifecycle phases presented by Kimball *et al.* (1998).

4.2.1 Agile system development methodologies (ASDMs)

The ASDMs investigated in chapter 2 has shared characteristics with regard to their suitability for data warehouse usage. These shared characteristics and their suitability for data warehousing is discussed in this section according to the phases of the data warehouse lifecycle proposed by Kimball *et al.* (1998).

4.2.1.1 Collecting requirements

Kimball *et al.* (1998) follows a requirements-driven methodology, i.e. requirements should be collected from trusted sources before data warehouse development takes place, providing an organisation with readiness for a data warehouse project.

Suitable Characteristics

It would thus be possible to use ASDMs that follow the same approach, where customer satisfaction is one of the main concerns. Although different ways to collect requirements exist, Kimball *et al.* (1998:97) suggest that interviews or facilitated sessions are conducted in order to collect significant information to develop a data warehouse of value to the business users. Interviews, although time consuming, lead to detailed data, as well as participation from different levels in the organisation. High-level requirements are given during interviews that will be the primary requirements to complete, after which extra requirements can be added by the users as development progresses. This emphasises the fact that high-level requirements are frozen early in the project to allow detailed investigation procedures, done by the team, to determine the consequences of the primary identified requirements (one of the values of agile processes).

Using ASDMs in data warehouse development will ensure early customer identification, resulting in early customer involvement and earlier requirements identification, making users actively involved throughout the development process (one of the core values of agile processes). Furthermore when using ASDMs, any means possible can be used to collect requirements, including techniques such as facilitated sessions and interviews explained by Kimball *et al.* (1998:97).

ASDMs make use of frequent incremental delivery, which leads to continuous customer involvement that result in feedback on the implemented requirements, emphasising the fact that frequent delivery has the highest priority (one of the core values of agile processes). Furthermore, ASDMs promote the availability of users to keep developers on track to satisfy requirements. This emphasises one value of agile process where, "users must be actively involved throughout the development process" (Hislop *et al.*, 2002:177).

Unsuitable Characteristics

ASDMs do not explicitly identify techniques that can be used during requirements collection. Furthermore, ASDMs primary concern is not on collecting requirements, they rather focus more on the development process by managing the process and constructing the product that will satisfy user requirements.

4.2.1.2 Data modelling

After collecting requirements, Kimball's approach suggests these requirements be modelled into diagrams. The diagram suggested by Kimball *et al.* (1998:194-199) is a star-schema, representing a dimensional model or data mart.

Suitable Characteristics

ASDMs do not specify which data model should be used to model

requirements, as ASDMs are process models that mainly focus on customer involvement and an iterative development process, not a data modelling technique. However, ASDMs do state that projects have different characteristics that must be accounted for, and during a data warehousing project a data model(s) is required. This will allow the team and users to empower themselves to make their own decisions, by using a star-schema to, model requirements, without the explicit approval of higher management (one of the values of agile processes). It would consequently be possible to use a star-schema to model requirements when ASDMs are considered for the development of a data warehouse using Kimball's star-schema.

Star-schemas present two strengths that can make them applicable when ASDMs are used in data warehouse development. Kimball *et al.* (1998:148) explains the first strength, where the predictive framework of a star-schema "withstands unexpected changes in user behaviour". The second strength of using a star-schemas, is that a star-schema "is gracefully extensible to accommodate unexpected new data elements and new designs" (Kimball et al., 1998:148). Graceful changeability has the advantage where no reporting or query tools need to be reprogrammed to accommodate the changes that was made by users or developers. In other words "old applications can continue to run without yielding different results" (Kimball et al., 1998:148). Thus, the graceful changeability of Kimball's star-schemas makes it applicable for all ASDMs to be used during the data modelling phase, as requirements can be added after development took place.

Unsuitable Characteristics

ASDMs do not provide explanations of star-schemas and ERDs as data modelling techniques. However, experience proves that system development is affected and limited by the "characteristics of the project, the people in the team and the organisation in which it works" (Lindstrom & Jeffries, 2004:50).

4.2.1.3 Data staging

During data staging, data is extracted from the data source system, transformed according to data warehouse standards and cleansed before loading it into the data warehouse. The 10 steps performed to clean data during data staging are explained in par. 3.5.5.

Suitable Characteristics

Kimball *et al.* (1998:630) state that the loading process can be done iteratively. Data staging (ETL-process) can be done in iterations when using an ASDM as an SDM.

The ETL-process of Kimball's approach can be viewed as the development process of every ASDM. ASDMs imply that team members may use their own tools and techniques to get the job done that empowers the team (one of the values of agile processes). Kimball *et al.* (1998:612) also agree that tools can be used to clean the data during the data staging process (i.e. step two of the data staging process).

ASDMs focus on developing a project in increments. Each individual increment can be completed iteratively so that altered and new requirements can be satisfied during the current increment. The idea of developing in increments can be applied during data warehouse development, i.e. the collection of data marts, where every data mart can be regarded as an increment or a set of increments (sub-data marts) that must be developed. Each data mart is developed and deployed separately. Each data mart can then be developed and delivered (if the data mart is very big) incrementally and rapidly, emphasising the fact that rapid iterations and incremental delivery are key to converging on acceptable business solutions (one of the values of agile processes). New and altered requirements can be added as development progresses for a specific data mart, which emphasises that reconstruction of previous data mart increments and iterations must be possible (one of the

values of agile processes). This approach will result in a data mart (data warehouse) that is up-to-date where most user and technical requirements are fulfilled.

With this said, the ETL-process for every data mart can be divided into iterations (or increments if a large data warehousing project is developed, where the extraction, transformation and loading processes are viewed as increments that can be done iteratively). This means that the extraction, transformation and loading process is completed iteratively to ensure that altered technical and user requirements are met in every increment. New and altered requirements can always be added in iterations as the data marts evolve.

The correct data must be extracted, transformed, and loaded, when using ASDMs to meet requirements in a data warehousing project.

Users can be either on-site or partially available (depending on how requirements are collected) while data staging takes place to assist developers in extracting, transforming and loading the correct data that will satisfy the user and technical requirements.

Unsuitable Characteristics

ASDMs do not list tools that can be used during data staging, which has an influence on its suitability within the data staging phase of Kimball's approach.

4.2.1.4 Data access and deployment

Access applications are created for business users in a manner that enables them to locate the necessary data as speedily and easily as possible in order to analyse the data to meet the users expectations. OLAP is used to create applications where users can execute queries on the data warehouse. Before deploying the data warehouse, alpha and beta tests must be completed before

the data warehouse is made available generally. Testing is not only performed as part of implementation, it is interacted throughout the development lifecycle where ASDMs are applied to data warehouse development (one of the values of agile processes).

Deployment takes place when the data warehouse has been completed. At this stage, end users are able to use the data warehouse and find it to be of value, and to evaluate whether developers made the right decisions during development. Users must be educated to gain maximum value from the data warehouse. After successful deployment, by evaluating data mart deliverables to determine its acceptable use (one of the values of agile processes), it is important to manage and maintain the data warehouse and prepare it for possible growth.

Suitable Characteristics

ASDMs show attractive characteristics in the implementation of a data warehousing project. Incremental (sub-data mart) delivery to a data mart and deployment of a data mart as a whole in data warehousing is key to converging an acceptable business solution, the latter being one of the nine principles reflecting the core values of agile processes.

Using ASDMs in data warehouse development and deployment, a small release can be viewed as a data mart that is delivered or implemented, becoming part of the main data warehouse. Before a data mart is implemented into the main data warehouse, it must be tested to identify whether requirements are met or not.

During the implementation phase of all ASDMs, the system is implemented and the operation of the system, or data warehouse for a data warehousing project, is transferred to the users. The users are trained to use the system effectively. If implementation occurs over a period in a data warehousing project, it may

also be done in increments. If new requirements are identified after a data mart have been deployed; it can be added by a simple iteration, meaning that backtracking or reconstruction of previous versions of data mart development must be possible (one of the core values of agile processes). The data warehouse will then be fully deployed if all increments (data marts) are successfully implemented for customer use. The data warehouse will only be a success if there is cooperation and collaboration between all team members and users (one of the core values of agile processes).

Unsuitable characteristics

ASDMs do not specify a specific way in which data should be accessed, which reporting tools should be used, in which ways users should be educated to use the data warehouse effectively or how a data warehouse should be managed and maintained to prepare it for possible growth. These are disadvantages of all seven ASDMs.

The general discussion on the suitability of ASDMs to data warehousing is now followed by individual explanations of how the unique characteristics of each of the seven chosen ASDMs can be applied on Kimball's data warehousing approach.

4.2.2 Dynamic Systems Development Methodology (DSDM)

This discussion of the suitability of the specific features of DSDM will be done according to the lifecycle phases of Kimball *et al.* (1998).

4.2.2.1 Collecting requirements

DSDM state that any means possible can be used to collect requirements. DSDM mainly focuses in keeping time and resources fixed while adjusting functionality accordingly, and keeping requirements in mind (Abrahamsson *et al.*, 2002:63).

Suitable Characteristics

Because DSDM recognizes requirement collection as important during the business study where requirements are collected using facilitated workshops, it would be possible to use DSDM during the collecting requirements phase of Kimball's approach, as Kimball *et al.* (1998) also use facilitated workshops. DSDM uses a technique called time boxing, to handle flexibility of requirements.

As an output of the business study, early client identification ensures early customer involvement (Kimball *et al.*, 1998:97; Abrahamsson *et al.*, 2002:65), and earlier requirement identification.

What makes DSDM even more suitable is that it is a people-oriented methodology, i.e. the requirements given by users are viewed as important.

Unsuitable Characteristics

DSDM, however, also has characteristics that make it unsuitable for collecting requirements during data warehouse development. One problem is that the primary focus is not collecting requirements. Furthermore, single interviews are not discussed. This implies that no interviews are scheduled or sequenced, and no kick-off meetings are held.

In addition, DSDM uses prototypes to capture information rather than detailed documentation, while Kimball *et al.* (1998) promote the use of documentation. (Highsmith, 2002a:8).

4.2.2.2 Data modelling

The DSDM Consortium (2003) states that DSDM can be used in a data warehousing project. This includes the use of data modelling techniques such

as star-schemas and ERDs.

Suitable characteristics

According to the DSDM Consortium (2003), DSDM products include a logical data model that contains a star or snowflake schema, or cube definitions focusing on the semantic integrity of the information presented to the user.

A star-schema represents a single business process. Because of the business study, the business area is defined by describing the affected business processes. These processes can be modelled into star-schemas when using Kimball's approach.

During the functional model iteration phase, the identified business process and requirements can be modelled by developing dimensional models (i.e. fact and dimension tables) that will satisfy and diagram user requests and requirements. During the design and build iteration the fact and dimension tables (data mart) can be built and implemented iteratively.

Avison and Fitzgerald (2003:286) suggest a modelling theme (star-schema or ERDs) to be used during system (data warehouse) development.

Unsuitable characteristics

Although the DSDM Consortium (2003) states that DSDM products include star-schemas, a star-schema (data modelling technique) is not mentioned in the six core techniques of DSDM. Authors of their methodologies will usually state that their methodologies will work in all kinds of projects, including modelling requirements into diagrams (star-schemas), as is the case with the DSDM Consortium (2003).

4.2.2.3 Data staging

DSDM provides several suitable characteristics where data staging can be implemented using Kimball's approach.

Suitable characteristics

The ETL-process is incremental, i.e. requirements can be updated and added as development takes place. In addition, the functional model iteration phase can be combined with step 1 (high-level plan) of Kimball's data staging process by building a schematic format (on one page) containing indications as to where the data has originated as well as the requirements identified in the business study phase. The schematic format can be a source-to-target map in a data warehousing project. A source-to-target map is a technique used by Kimball *et al.* (1998) to diagram where the data in a star-schema originates from using the ERD tables of the operational system.

The ETL-process can be done iteratively during the design and build iteration phase of DSDM during which the requirements are designed and satisfied.

Also advantageous is the selection of tools that can be used (Avison and Fitzgerald, 2003:286, 331):

- Oracle (Oracle Database Management System, Designer/2000, Developer)
- Business process modelling
- UML profile
- Graphical simulations of UML designs
- Database modelling and code generation
- Design patterns and optional component-based techniques
- Scalable enterprise repository
- Intelligent document generation
- Traceability and impact analysis
- Java, Visual Basic, C++ code synchronization
- Integration with a range of other tools

The techniques that can be used are (Avison and Fitzgerald, 2003:154,286):

- UML
- Joint application development (JAD)

Unsuitable characteristics

It is a disadvantage that not all of the identified tools by DSDM are suitable for data staging during data warehouse development.

4.2.2.4 Data access and deployment

During the implementation phase of the DSDM lifecycle (fig 2.2), the system is implemented and the operation of the system, or data warehouse for a data warehousing project, is transferred to the users (DSDM Consortium, 2005).

Suitable characteristics

This implementation phase is repeated for every data mart in a data warehouse project until all data marts are deployed as a whole, forming a data warehouse.

During the implementation phase, the incremental review document can be used to discuss the state of the system by testing it, and preparing it for further growth and maintenance. Output of implementation is a user manual that can train the users to use the implemented data warehouse effectively. The project review document can be used to know what to maintain in the developed data warehouse, as well as what preparations should be made for possible growth.

During the post-project phase, the system is thoroughly tested and maintained to ensure that the deployed data warehouse continues to be valuable and of use. The output of the design and building phase is a tested system (data warehouse) that meets at least the most important requirements set by users. A big advantage of DSDM is that the system is constantly being tested throughout

the development process.

Incremental development and deployment is used to complete, combine, and test prototypes of sufficient quality and to release them safely to users (DSDM Consortium, 2005).

Unsuitable characteristics

DSDM is more concerned with developing what the customer expects than with deploying the system or data warehouse in a predefined way.

4.2.3 Scrum

4.2.3.1 Collecting requirements

Using Scrum, any means possible can be used to collect requirements from software developers, users, and experts. Techniques such as facilitated sessions and interviews can thus be used in a Scrum project, therefore making it possible to use Scrum during the collecting requirements phase of Kimball's approach.

Suitable characteristics

The fact that Scrum is requirements driven (i.e. requirements are viewed as very important throughout the Scrum lifecycle), makes it suitable within the data warehouse development process, because Kimball *et al.* (1998) also follows a requirements-driven methodology where requirements can change. These changeable requirements can be managed and implemented within the evolving system using Scrum.

During the pre-sprint planning phase, requirements are identified and extracted from the prioritized product backlog to the sprint backlog to be completed during the next sprint (Cohen et al., 2003:14).

The requirements set by users, managers and experts during interviews are satisfied during a sprint (30 days). With Scrum, 15 minute meetings every day ensure that requirements are met and cause new requirements to be discovered. Another suitable characteristic is that after each sprint a post-sprint meeting is held that may result in the identification of new requirements to improve the system or data warehouse. The product backlog must be constantly updated and prioritized as new requirements are identified.

Unsuitable characteristics

The fact that Scrum focuses more on team empowerment (Huijbers et. al., 2004:19) than collecting requirements, can make it unsuitable for collecting requirements during a data warehousing project. This focus implies that communication is more important than collecting requirements. In addition, Scrum focuses on the development process by managing the process and constructing the product.

4.2.3.2 Data modelling

The data warehouse lifecycle components of Kimball's approach can be incorporated into the building blocks of Scrum, firstly by using a sprint to model requirements into an acceptable star-schema.

Suitable characteristics

During the first sprint the key pieces (building blocks), including the initial system framework (i.e. star-schemas in the case of data warehouses), technological requirements (i.e. programmes used to develop a data warehouse), and business functionality are estimated and completed. Even more advantageous is that because requirements have a tendency to change, updated requirements can be modelled into the star-schemas using iterative development of Scrum.

Scrum has been found successful in exporting and importing data in a data warehouse during a project in a university environment (Mahnic & Drnovscek, 2005:1).

Unsuitable characteristics

Scrum does not explain star-schemas or other data modelling techniques to be used during project (data warehouse) development, which diminishes its suitability in the data modelling phase.

4.2.3.3 Data staging

In order to give users what they ask for in a data warehousing project, the correct data must be extracted, transformed and loaded. An aspect that distinguishes Scrum from other ASDMs is that a Scrum meeting is held every day and development takes place in sprints. A sprint in a data warehouse project can be seen as an iteration of an increment (sub-data mart) in large projects, or an increment of a data mart in a small project. The data mart can thus be defined in sprints, i.e. a specified workload is completed during each iteration (sprint) to deliver a sub-data mart that becomes part of the data mart. Scrum doesn't specify whether tools or manual written data staging programs can be used during data staging, although Scrum recognises that every project has its own characteristics and degree of uncertainty. Thus, it would be possible to use data staging tools or manually written programs to clean data during the ETL-process of a data warehousing project. Scrum can be used during the ETL-process in several ways.

Suitable characteristics

An advantage of Scrum is that it accepts the unpredictability of the development process. This means the ETL-process can be used during the development of a data warehousing project.

System development using Scrum involves requirements, resources, technology, and time constraints. In this instance, technology can be viewed as data staging tools that can simplify the ETL-process.

Abrahamsson *et al.* (2002:29) explains the product backlog as part of the planning phase, i.e. as a sub-phase of the pre-game phase, where team, resources, controlling issues, and tools are defined. These defined tools, are the tools that can be used during data staging to extract, clean, and load the data.

By using Scrum during a data warehousing project, the requirements can be extracted from the product backlog to the sprint backlog to be completed and transformed during the next sprint. After successful completion of the sprint, the requirements that have been met can be loaded, to become part of the increment (sub-data mart) that is being developed in large data warehousing projects. The sprint backlog and sprints are part of the development process (Abrahamsson et al., 2002:28), thus the ETL-process can be executed using the product backlog and sprints.

To keep track of the ETL-process with Scrum, short 15 minute meetings can be held daily to identify and solve problems during the development of a demanding data warehousing project. Post-sprint meetings can be held to analyze whether the iterative (in small projects) or incremental (in large projects, where every increment (extract, transform or load) is developed iteratively) ETL-process has been successful. If it is successful, the next sprint is done by extracting the next requirements from the product backlog. If not, the sprint(s) is repeated.

Scrum can be adopted in existing and new projects (Schwaber & Beedle, 2002:59).

Unsuitable characteristics

However, Scrum does not list tools as DSDM that can be used during data staging, which has an influence on its suitability within the data staging phase of Kimball's approach.

4.2.3.4 Data access and deployment

Scrum does not only have the ability to design a new project, but it can also be adopted in existing projects. As explained in par. 4.2.3.3, the characteristic that distinguishes Scrum from other ASDMs, is that development is done in sprints. If requirements change or new requirements need to be added, it can be done using a simple iteration.

Suitable characteristics

Scrum has a built-in phase where implementation and maintenance takes place, called the post-game phase. The architecture phase, which is a sub-phase of the pre-game phase, includes the changes, and the problems these changes may cause in implementing the product backlog if the implemented system (data warehouse) requires enhancement (Abrahamsson *et al.*, 2002:29).

Planning, yet another sub-phase of the pre-game phase, "includes the definition of the system being developed, project team, tools and other resources, risk assessment and controlling issues, training needs and verification management approval" (Abrahamsson *et al.*, 2002:29). Planning can thus be used to plan a training program for the users.

The post-sprint meeting as described in par. 4.2.3.3 can be used to identify whether incremental implemented design meets the expectations of the users. One of the key principles of Scrum is that the system must be constantly tested and documented.

Unsuitable characteristics

Scrum is not clear about a specific way in which a data warehouse should be implemented, because it is more focused on the development and success of development than on the implementation process.

4.2.4 Extreme Programming (XP)

4.2.4.1 Collecting requirements

XP mainly uses story cards to collect requirements. When facilitated sessions or interviews (any data collection technique can be used by XP) are used in an XP project, the requirements mentioned can be written on story cards to help developers apply their minds to what is expected from the data warehouse. The mentioned requirements can also be prioritized much easier using story cards.

Suitable characteristics

XP's suitability in the collecting requirements phase of Kimball's data warehouse firstly lies in the fact that requirements can be collected with story cards.

XP regards requirements as very important, as they are addressed in the four values (communication, simplicity, feedback, and courage) and four activities (coding, testing, listening, and designing) of the methodology. This ASDM focuses on communication between the team and its users to gather requirements using techniques such as interviewing or facilitated sessions.

Simplicity requires that the simplest system (or data warehouse) that will satisfy the requirements set by business users and sponsors be developed.

During feedback, users define their requirements on story cards and developers estimate the correct approach for immediate feedback to users on

the work they will do to satisfy these customer requirements.

Developers furthermore need courage to develop a data warehouse that fulfil most requirements, because the process consumes time and acquires intellectual skills.

Another characteristic that makes XP suitable is that pair programming is used to generate code, code that does what it is designed to do in order to satisfy user requirements. Developers should listen to clients to know which requirements should be tested (Hislop *et al.*, 2002:174).

Story cards are used to simplify listening so designers can design the specified user requirements. Designing is a continuous activity of incorporating new requirements into the evolving and already existing system (data warehouse). Requirements are one of the main building blocks within the XP lifecycle and twelve XP practices, as they are integrated throughout the lifecycle.

Lastly, the XP methodology is suitable for data warehousing because users are always on-site to keep developers on track to satisfy requirements. These requirements are visible throughout the rhythm of an XP project, i.e. where requirements are written on story cards, discussed, implemented and tested.

Unsuitable characteristics

Kimball *et al.* (1998) mainly collect requirements using facilitated sessions or interviews, while XP uses story cards that must be given for the users to write down these requirements.

4.2.4.2 Data modelling

XP's twelve core practices have the potential to incorporate requirements gathered into a data model during a data warehouse project.

Suitable characteristics

Advantages of XP include that "simple design" (one of the twelve practices of XP) can be accomplished by using star-schemas were requirements could be modelled into an understandable format for users and developers. "Refactoring", also one of the twelve practices of XP, can be accomplished through using star-schemas to remove field and record duplication, improving communication and by adding flexibility to a difficult design.

Furthermore, a star-schema can be used to organize and model requirements in an organized fashion during the "planning game" (another practice of XP). Because users are on-site the whole time (also one of the twelve XP practices), new and adjusted requirements can be added to the dimensional models (star-schemas).

Unsuitable characteristics

A disadvantage is that XP cannot be adopted in all kinds of projects. Aveling (2004:98) interviewed four companies that struggled and failed to implement some of the practices of XP. Aveling (2004:94) states that "most existing studies are the post-hoc assessments of the authors' adoption of XP", and further explains that "most organisations adopt XP only partially".

4.2.4.3 Data staging

XP shows great potential to work during the development of a data warehouse, and especially during the ETL-process. In fig. 2.6, XP's lifecycle shows how requirements (story cards) are satisfied by programming, testing and releasing iteratively. XP uses pair programming where errors can easily be identified and corrected during the transformation and data cleansing process. Graziano (2005:5) emphasises that pair programming can be used during the ETL-process, especially with Oracle Warehouse Builder and the input of PL/SQL code into Oracle Designer. XP is suitable for use during the ETL-process in various areas.

Suitable characteristics

Developing a data warehousing project using XP is possible using an incremental (or iterative - for small projects) ETL-process and "continued integration" (one of the twelve core practices of XP). "Coding", one of the four activities of XP, is seen as a learning activity where the requirements are coded and transformed to satisfy users' needs in a data warehousing project.

"Designing", another of the four activities of XP, is a continues activity of incorporating new designs of the ETL-process into the existing system (data warehouse). Development is incremental, i.e. development takes place in "small releases" (one of the twelve core practices of XP), one sub-data mart at a time, until the data mart is deployed as a whole becoming part of the existing data warehouse.

The "metaphor", another core practice of XP, guides the design, meeting user requirements by loading data that is of value. The core practice, "testing" ensures that the ETL-process is followed correctly. It entails that programmers ensure that the code that has been written, does what it is designed to do.

During the refactoring core practice, duplication is removed by using data staging tools. Another advantage is that code quality can be increased trough using "collective ownership" (Lindstrom & Jeffries, 2004:49) when developing the GUI (Graphical User Interface) of the data warehouse.

"Coding standards", another core practice of XP, ensures that the transformation process in a data warehousing project remains the same throughout the project.

Unsuitable characteristics

There are aspects of XP that could make it unsuitable for use in the ETL-

process of a data warehousing project. "Collective ownership" (core practice of XP) may delay the data staging process if new developed data staging code is changed by inexperienced programmers.

4.2.4.4 Data access and deployment

XP delivers the system in small releases or increments. A "small release" can be seen as logical grouping of tasks or requirements (increment) that should be delivered iteratively to a sub-data mart that becomes part of a data mart in a large data warehousing project, where after the data mart is deployed as a whole to become part of an existing data warehouse. Before a small release (sub-data mart) is implemented, it must be tested to identify whether requirements are met or not. In small projects, using XP during data warehouse development and deployment, a "small release" can be viewed as a data mart that is deployed, becoming part of the main data warehouse.

Suitable Characteristics

XP shows acceptable characteristics for implementing a data warehouse in the various ways. Simplicity, one of the four values of XP, can be achieved by keeping the system easy accessible for users, using maintenance, and by upgrading the data warehouse regularly.

One of the fifteen principles that support XP's values emphasizes "teaching" and "learning". This entails teaching users to use the data warehouse effectively. "Testing", one of the four activities and twelve practices of XP, is a continuous activity that takes place throughout the development process of XP and data warehousing. Another advantage is that in XP, "designing", which is also a basic activity of the methodology, is a continuous activity of incorporating new requirements into the evolving system or data marts.

By "listening" (an XP value) to user expectations, designers will know how to

manage the data warehouse to deploy an effective data warehouse.

Every time a task has been completed, it is integrated into the system or data mart. Then tests are run. The data marts must pass these tests for the changes in code to be accepted (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12; Lindstrom & Jeffries, 2004:48). After this is done, users of the data warehouse can be trained to gain easy and valuable access to the data warehouse.

Unsuitable characteristics

No specific way of implementation is specified by XP. This is necessary when employees have to continue working on the old existing system.

Kimball *et al.* (1998) promotes documentation and a user manual, while XP is "lite" and prefers that documentation be cut away. Furthermore, XP believes a user should be trained to use the data warehouse effectively instead of giving the user a user manual.

4.2.5 Feature Driven Development (FDD)

4.2.5.1 Collecting requirements

A feature is the primary component that drives development in an FDD project. Features in FDD can be viewed as requirements gained from trusted sources that must be fulfilled in the course of a project. FDD does not explicitly state which techniques should be used during the process of collecting requirements, as long as the requirements come from trusted sources. Thus, it would be possible to use facilitated sessions or interviews to gather requirements (features) in a data warehousing project. These features are then prioritized in a features list and grouped in feature sets to be developed.

Graziano (2005:4) states that a feature set may be equivalent to a subject area or data mart, or a sub-division thereof in a data warehousing project. Using an

agile approach, the data warehouse can then be developed in sub-feature sets or sub-subject areas (sub-data marts) in large data warehousing projects (Graziano, 2005:4). This holds that a data mart (in small projects) or sub-data marts (in large projects) can be developed in iterations. The sub-data marts can then be delivered incrementally to the data mart, where after the data mart is deployed as a whole, instead of modelling and developing the whole data warehouse all at once.

Suitable characteristics

When it comes to suitability, FDD has several positive characteristics. FDD's features contain requirements written in a language understandable to all parties associated with the project (like XP's story cards).

As in Kimball's data warehouse, FDD firstly identifies requirements and develops an overall model where users, sponsors and other role-players know the specified requirements. During the process of "building a feature list", the features (requirements) are grouped together, each group representing a specific domain. The collected feature list is then prioritized and assigned to chief programmers responsible for meeting the assigned requirements.

During the "design by feature and build by feature" phase a feature set (data mart) is selected, designed and tested in increments (data marts) before becoming part of the main system (data warehouse).

Unsuitable characteristics

FDD primarily focuses on the design and building phases of the software development process (Abrahamsson et al., 2002:47), and not on the techniques for collecting requirements. This could make the methodology unsuitable for collecting requirements.

4.2.5.2 Data modelling

FDD as explained by Graziano (2005:4) can be used during the modelling as well as the data staging phase of a data warehousing project. The data model of a data warehouse can be modelled in some key areas of the FDD ASDM. During the development of an overall model a star-schema can be constructed in which all the requirements are encapsulated in the data model(s).

Suitable characteristics

FDD shows potential in several areas. During "domain object modelling" (a best practice of FDD), a framework (star-schema) is developed to which features can be added. One of the values that work best for developing a project using FDD explains that FDD works for projects that have the ability to grow, similar to data warehouses.

During the "build a feature list" phase, the features (requirements) can be identified and represented through using star-schemas in data warehousing projects. Star-schemas can be designed by collecting and diagramming a list of features (requirements) into design packages. This forms part of the "plan by feature" phase. During the "design and built by feature" phase the selected features (requirements) are planned in more detail by developing the final star-schema(s).

FDD has a high success rate in large projects (data warehousing projects), if diverse talent is available (Highsmith, 2002a:6). FDD is also suitable for new projects, projects in need of code upgrading, and projects that require the development of a second version (Abrahamsson *et al.*, 2002:54). This could include data warehousing projects.

Unsuitable characteristics

FDD could be unsuitable because it primarily focuses on the design and building phases of the development process and does not require a specific

data model (star-schema) to succeed (Abrahamsson *et al.*, 2002:47).

4.2.5.3 Data staging

By using FDD the data warehouse will be developed in feature sets, or, as explained in par. 4.2.5.1, in data marts or sub-data marts (increments) in large projects. When using FDD, the features can be selected from the feature list to be planned, built, tested, and integrated as increments in the already existing data warehouse.

Team huddles, also known as morning roll calls, is a "concept that can definitely be applied to data warehouse projects ... it keeps everybody in sync" (Graziano, 2005:5). These team huddles, like the daily meetings of Scrum, keep developers op to date on the progress of development and the data staging process as the project progresses.

Suitable characteristics

By using FDD, a data warehousing project can be developed through an incremental ETL-process. FDD focuses on the design and building phases of the development process (Abrahamsson *et al.*, 2002:47), contributing to the fact that it is an advantage when FDD is used to clean data during a data warehousing project.

The ETL-process can be done during the "design by feature and build by feature" phases where the features (requirements) are planned, built, tested, and iteratively integrated within the existing system.

The best practices of FDD should be composed into features, planned – i.e. decide what should be extracted, transformed and loaded – designed and coded.

One of the core values of FFD is that the process must be "logical". The ETL-

process is logical, because it utilizes extraction transformation and loading. The ETL-process has no "process pride" (core value of FDD), meaning that developers other than the authors have proven FDD to be effective.

FDD has a high success rate in the development of new as well as large projects (Highsmith, 2002a:6).

Unsuitable characteristics

FDD does not mention any tools or techniques that can be used during data staging of a data warehousing project, which puts it at a disadvantage.

4.2.5.4 Data access and deployment

Using FDD, features can be iteratively designed, tested, and integrated to deliver sub-data marts incrementally to a data mart, where the data mart can be deployed to become part of the existing data warehouse in large data warehousing projects.

Suitable Characteristics

FDD can be used for implementing a data warehouse, because "reporting progress", one of the best practices of FDD, helps to maintain the system and allows designers to produce a data warehouse of value to users.

FDD encapsulates best practices and incremental development to manage and monitor the development process after deployment (Abrahamsson *et al.*, 2002:47; Hislop *et al.*, 2002:175). This can be applied to data warehousing.

After successful completion of each iteration (i.e. the ETL-process and design), including coding, testing and iteration, the current iteration becomes part of a sub-feature set or increment (can be one of three increments: extract, transform or load) in the ETL-process of a large data warehousing project.

During "design by feature and build by feature", the features (requirements) are planned in more detail, built, tested, and integrated incrementally into the system. This can be applied to data warehouse development where sub-data marts are delivered incrementally to a growing data mart.

Unsuitable characteristics

FDD primarily focuses on the design and building phases of the development process (Abrahamsson *et al.*, 2002:47), and not on implementation. It does not specify a method of implementation especially during the changeover period when employees still have to use the existing system.

4.2.6 Crystal Clear (CC)

4.2.6.1 Collecting requirements

CC focuses on reducing documentation and promoting face-to-face communication. Using osmotic communication (i.e. where developers work closely together), developers encourage each other to fulfil the requirements, and to make sure everyone fully understands what is expected from the project.

Suitable Characteristics

CC has several advantages, which makes it suitable for use within the collecting requirements phase of Kimball's data warehouse. Crystal ASDMs focus primarily on communication and people, i.e. requirements gained from communication between the team and business users are seen as very important.

In addition, Crystal ASDMs aims to address different kinds of project requirements with different kinds of Crystal ASDMs. "Frequent delivery", one of the seven CC properties, causes continuous customer involvement resulting in feedback on the implemented requirements. Requirements that have not been

met, are identified and the project team is given time to eliminate the deficiencies.

"Face-to-face communication" in CC is advantageous because it leads team members to understand each other and the requirements that have to be satisfied.

Unsuitable characteristics

The fact that CC may be too small for a large data warehousing project, could constitute a disadvantage in the collecting requirements phase. CC is designed for small projects where team members share the same office space to enhance communication and help each other understand requirements better.

4.2.6.2 Data modelling

CC mainly consists of seven properties to enhance communication and to deploy a satisfactory product. CC does not define any techniques for modelling requirements into diagrams such as Kimball's star-schema. It does, however, state that projects have different characteristics that must be accounted for, and during a data warehousing project a data model(s) is required. It would consequently be possible to use a star-schema to model requirements if CC is considered for the development of a data warehouse using Kimball's star-schema.

Suitable characteristics

"Focus", one of the seven properties of CC, can be established by using ERDs or star-schemas to keep the focus on the objectives (satifying collected requirements) which will then be completed much quicker. Team members can use their own techniques (star-schemas) and tools to satisfy the seven properties of CC, techniques that may include star-schemas as every project (data warehousing project) has its own characteristics.

Unsuitable characteristics

CC values "properties over techniques" (Huijbers *et al.*, 2004:21). These techniques, including ERDs and star-schemas, are very important for data warehouse development. Consequently, CC could be unsuitable in the data modelling phase cycle.

4.2.6.3 Data staging

Examining the seven properties of CC, the methodology shows the ability to be used during the ETL-process of a data warehousing project.

Suitable characteristics

Properties such as "frequent delivery" and "incremental (iterative) delivery" emphasise that it may be possible to use these properties during the ETL-process. To deliver frequently and incrementally, developers should extract, transform, and load data iteratively in a data warehousing project. This agrees with the suggestion of Kimball *et al.* (1998:630) that loading should be done iteratively.

Although CC does not state whether tools or newly developed programs can be used to clean data, it accepts the uncertain nature of projects and suggests that developers with enough experience use their own techniques and tools to get the job done. It thus harmonises with Kimball *et al.* (1998:612) who suggest that tools are used during data staging (step 2 of the data staging process).

Unsuitable characteristics

As explained in par 4.2.6.2, CC values "properties over techniques" (Huijbers *et al.*, 2004:21). CC also does not list any tools that can be used during data staging – a disadvantage when using the methodology during data staging.

4.2.6.4 Data access and deployment

CC emphasises "frequent delivery". The only way of delivering frequently is by delivering incrementally. It would thus be possible during a project to deliver the data warehouse frequently and incrementally.

Suitable characteristics

Using "frequent delivery" and delivering on a regular basis (on of the six standards of CC), working code are tested and implemented into the system. During "reflective improvement" (property of CC), the system is monitored and maintained to identify flaws while development and implementation continues.

Furthermore, CC has a technical environment where testing and controlling tasks, e.g. making backups and merging changes, are executed during automated testing, configuration management and frequent deliveries.

One of the six standards of CC requires a release to be tested by two users before implementation within the existing system or data warehouse.

Unsuitable characteristics

As explained for all ASDMs, there is no specified manner of how data should be accessed and implemented or which reporting tools should be used by CC.

4.2.7 Adaptive Software Development (ASD)

4.2.7.1 Collecting requirements

ASD uses JAD sessions to gather requirements. This entails that developers and users meet to discuss ideas and product deliverables, and to enhance communication. ASD does not state that facilitated sessions or interviews can be used in conjunction with JAD sessions. During JAD sessions, the users are brought into the development process as active participants (as with all ASDMs that focus on user involvement), while during interviews and facilitated sessions

requirements are gathered before development takes place. It would thus be possible to use a facilitated session or interviews before development takes place to identify primary requirements, and to update requirements while development takes place by implementing JAD sessions with users in a data warehouse project where Kimball's approach is used.

Suitable characteristics

JAD sessions has an advantage where the collected requirements and ideas can be brainstormed. During the "speculation" phase of ASD the uncertain nature of complex problems should be recognized and developers should determine whether the requirements could be satisfied.

During the project, in the "initiation step" (i.e. the first step of the adaptive cycle speculation) requirements are identified and collected, after which the "project time box" (duration of data mart) is determined based on the requirements set by users. Teams can effectively work together (collaborate) if communication is good and if they know what to do (requirements) and how to do it (satisfy the requirements).

Furthermore, review practices ensure that requirements are met in the learning phase. The "learning loop" can be used to identify new requirements in a data warehousing project to be implemented within the data warehouse during the learning phase.

One of the characteristics of an adaptive lifecycle is time boxing, where fixed delivery times for iterative cycles are estimated to deliver satisfied requirements (sub-data marts) and where requirements can be redefined.

A important advantage is that ASD can be used in large projects, such as a data warehousing project, to deploy a product that is up to date and meets the most requirements.

Unsuitable Characteristics

Because ASD identify only JAD sessions as a requirements collection technique, it could make this methodology unsuitable within the collecting requirements phase. Interviews and facilitated sessions are not mentioned by ASD that is suggested by Kimball *et al.* (1998:97).

4.2.7.2 Data modelling

Although ASD does not mention the use of star-schemas, ASD present areas where a star-schema can be incorporated into a data warehousing project.

Suitable characteristics

ASD focuses on results. These results can be achieved using star-schemas to model the requirements that should be satisfied in a data warehousing project. "Speculation", which is part of the adaptive lifecycle, leaves room for exploration. This may lead to the inclusion of star-schemas in a data warehouse project.

Using ASD for large projects, a large volume of information must be collected, analyzed, and applied to solve the problem. Star-schemas could be used in data warehousing projects for analysis and collection. During the "project initiation" phase, which is step 1 of adaptive cycle speculation, the requirements should be represented by using star-schemas in data warehousing projects.

Unsuitable characteristics

ASD does not explicitly state how data should be modelled as it is more concerned about collaborate teamwork and regarding every problem as a learning activity to create opportunity.

4.2.7.3 Data staging

ASD does not specify a tool that can be used during transformation, as ASD focuses mainly on collaboration, speculation, and learning activities. In using ASD as an ASDM for data warehousing, it is essential to incorporate the ETL-process.

Suitable characteristics

ASD focuses on results by meeting users' needs. In order to meet requirements, the correct data must be extracted, transformed, and loaded during a data warehousing project. In designing large and complex problems, a large volume of information must be collected (plan what to extract), analyzed (transformed) and applied (loaded) to solve the problem.

Developers should be able to adapt by focusing on "learning". The best way to learn is to transform data in a data warehousing project to see what the data can present to the users. Development is a learning process because developers learn from their mistakes.

During the adaptive lifecycle activities, "project initiation" and "adaptive lifecycle planning" (see fig. 2.12) can be replaced by planning what should be extracted to satisfy user requirements in a data warehousing project. The transformation process (see fig. 2.12) can replace "concurrent component engineering" during a data warehousing project. "Quality review", and "final Q/A and release" (see fig. 2.12) can be performed iteratively where new and changed requirements can be identified.

The ETL-process can be executed using "timeboxes" and "release cycles". In large data warehousing projects the ETL-process can be divided into increments (extract transform and load) where every ETL increment is developed in timeboxes (iterations) and delivered to the data mart in release cycles. The data mart can then be deployed as a whole in a data warehousing

project.

Using component based development, which is one of the characteristics of the adaptive lifecycle where a group of features to be developed are defined, the ETL-process can be done iteratively (in small project) by planning the requirements to be extracted, transformed and loaded to meet users' expectations.

Unsuitable characteristics

Managers are more worried about dealing with collaboration and concurrency than about the details of designing, coding and testing when using ASD, which affects its suitability.

4.2.7.4 Data access and deployment

Although ASD focuses more on collaboration, speculation, and learning activities, it would be possible to deploy a data warehouse, one data mart at a time.

Suitable Characteristics

Several suitable characteristics to deploy and maintain a data warehouse are notable. One of the four categories of lessons to be learned is the "project status". This will only be known if the data warehouse is tested, monitored, and status reports are generated.

"Final Q/A and release" (see fig. 2.12) can be replaced by deployment and data access in a data warehousing project. Maintenance and testing can be integrated throughout the "learning" phase of ASD where the technical quality of the project is reviewed. The review practice of providing visibility and feedback from the users, explains how to review and test the system to determine what users expect. Furthermore; the team's performance is

monitored.

The (data warehouse) project status can be reviewed by applying maintenance controls and tests on the existing system to determine the value of the data warehouse, and determine whether it meets user expectations.

The "learning loop" is gained from repeated quality reviews in fig. 2.12 (Abrahamsson *et al.*, 2002:70), using maintenance controls and tests to deliver a data warehouse of value.

Unsuitable characteristics

As explained in par. 4.2.7.3, managers are more concerned with dealing with collaboration and concurrency than with the details of designing, coding and testing, which influences ASD's suitability.

4.2.8 Lean Development (LD)

4.2.8.1 Collecting requirements

LD states in practice 2 (amplified learning) that developers can use their own techniques and digression to complete a specific phase, in this case: collecting requirements. Because developers can use any technique, it would be possible to collect requirements before development takes place through facilitated sessions or interviews, and to update these requirements using "feedback" (one of the 22 tools of LD) from users in a data warehouse project where Kimball's approach is used.

Suitable Characteristics

Several characteristics make LD suitable to collect requirements for a data warehousing project.

"Amplify learning", one of the seven principles of LD, uses "feedback", one of

LD's 22 tools, where the developers determine whether user requirements are satisfied. The principle of "delay commitment" causes users to delay their decisions until they have enough accurate information to supply the correct requirements to be implemented iteratively within the system (data warehouse). "Deliver fast" (fourth principle) means satisfying user requirements as quickly as possible.

During "build integrity in", another of the seven principles of LD, perceived integrity and conceptual integrity are achieved when users are satisfied by meeting their requirements.

Unsuitable characteristics

"Lean thinking" may cause some requirements to be overlooked in a data warehousing project. Lean thinking entails letting users delay their decisions about what they want, and when they ask for something, provide it so quickly that they do not have time to change their minds. The principle, "delay commitment" may be a disadvantage in large projects that must be completed in a specific period, because if developers wait for requirements, development is delayed. In most cases, users will be more decisive if they have a basis system (data warehouse, or data mart) to work on.

Another disadvantage is that the overall success of the project (see the whole) is more important to LD than the traditional sub-optimization of individual tasks, such as identifying requirements.

4.2.8.2 Data modelling

It would be an acceptable approach to use Kimball's star-schema when data is modelled into diagrams in a data warehouse where customer and requirement satisfaction is the primary priority, as is the case with LD.

Suitable characteristics

According to Poppendieck (2003:2), "Great designers understand that designs emerge as they develop a growing understanding of the problem." This means that star-schemas will emerge if Kimball's approach is used to develop a data warehouse, which is an advantage. Through empowerment, the team can make its own process designs (star-schemas), commitments, goals, and decisions. "Empowering the team" is one of the seven principles of LD.

Unsuitable characteristics

On the down side, no ERD or star-schema structure is mentioned as one of the 22 tools of LD. As explained in par. 4.2.8.1, the authors of LD state that LD is a management philosophy rather than a development process.

4.2.8.3 Data staging

The data staging process can be integrated in all of the seven LD principles in several areas. The principles are; eliminate waste, amplify learning, delay commitment, deliver fast, empower the team, build integrity in, and see the whole. Development of a data warehousing project can be done through an incremental ETL-process by using LD.

Suitable characteristics

The ETL-process can benefit from "lean thinking" and "delayed commitment", which entails that users' decisions are delayed so that they know exactly what they want. When they then ask for it, it is given to them so quickly they would not have time to change their minds.

During "elimination of waste", duplication fields and records can be removed using the data staging process in a data warehousing project. During the transformation process, a developer will "learn" when data is transformed to identify its capabilities to satisfy users.

LD has a choice of 22 tools. Because the ETL-process can be done iteratively, in small data warehousing projects, or incrementally, in large data warehousing projects where every increment is done iteratively, a data warehouse can be delivered fast by using the tool, "queuing theory" where "cycle" times are kept short (Steindl, 2004). Integrity can be built by loading data that is of value to users (i.e. giving users what they expect).

Another advantage of LD is that it has been proven successful in large telecommunication projects in Europe (Highsmith, 2002a).

Unsuitable characteristics

As explained in par. 4.2.8.1, lean thinking may cause development to be repeated because some requirements may not have been satisfied. None of the 22 tools focuses on the cleansing of the data during a data warehousing project.

4.2.8.4 Data access and deployment

LD does not explicitly explain how a data warehouse project should be deployed or maintained. It does however, specify that deliveries should be quick, i.e. where every sub-data mart (increment) is delivered to the data mart that will be deployed as a whole.

Suitable characteristics

LD focuses on the elimination of waste, e.g. duplicate data, by looking at the flow of value from request to implementation. (The elimination of waste is a principle of LD.) In "quick delivery", another of the seven principles, the goal is to create value as fast as possible by allowing no delays during testing, integration and deployment, similar to the way in which Kimball *et al.* (1998) promotes deployment of a data warehouse.

During "empowerment of the team", the whole team – including on-site and other users – can be trained to use the system or data warehouse effectively.

To achieve perceived integrity, the correct requirements should be tested and implemented after development to meet users' expectations.

A mature system serves users with speed, repeatedly and reliably (Poppendieck, 2003:3), and delivers a system or data warehouse (as a data warehouse is seen as a system in this study) of value to users.

Unsuitable characteristics

Implementing a data warehouse in one-third the time, a third of the budget and one third the defect rate is an almost impossible task, which makes LD unsuitable for a large data warehousing project.

4.3 Inmon's approach

Since the use of ASDMs are limited to certain phases of Inmon's approach the discussion of the suitability of ASDMs for Inmon's approach is done according to Inmon's development phases. Thus, in this section the shared characteristics and unique individual characteristics with regard to ASDMs suitability towards data warehouse development will be excluded for every phase of Inmon's approach (accept for the data access and deployment phase). Only the unique individual characteristics of the seven ASDMs will be explained for the data access and deployment phase of Inmon's approach.

4.3.1 Collecting requirements

Inmon (1996:144) states "requirements for a data warehouse cannot be known a priori". Although Inmon does not collect requirements before development takes place, requirements are identified after deployment takes place by using

the "feedback loop". During the "feedback loop", new and alternative requirements are identified by the DSS analyst who sends them to the data architect. The identified requirements are then implemented by the data architect into the existing data warehouse, in order to keep the data warehouse up to date and of value for users.

The seven ASDMs emphasise that requirements must be collected before development takes place, and new requirements added as the system evolves. Thus, it would be unwise to develop a framework for the seven ASDMs in this phase of Inmon's approach, because ASDMs rely on collecting requirements before system (data warehouse) development. Contrary to this, Inmon's approach relies on a "feedback loop" to identify requirements after deployment takes place.

Furthermore, review practices ensure that requirements are met in the learning phase. The learning loop is used to identify new requirements to be implemented within the system during the learning phase.

4.3.2 Data modelling

Inmon (1996:85) proposes the use of an ERD data model above Kimball's star-schema for developing a data warehouse. A data model can be developed by using the corporate model as a starting point. A corporate ERD consists of a combination of ERDs with every ERD reflecting the different views of people within an organisation.

ASDMs have the ability to deliver a system or data warehouse in a constantly changing environment. To deliver a system in such an environment, ASDMs suggest incremental and iterative development (see par. 2.3) whilst primarily focussing on communication and requirement satisfaction. Some ASDMs, e.g. DSDM, ASD, Scrum, and XP with it's twelve practices, follow a lifecycle, while others follow either a process (FDD), properties (CC), practices (XP) or principles (LD) to deploy a product (data warehouse) in which at least the most

important requirements have been met. Most ASDMs, e.g. LD, ASD and CC, specify that team members, under supervision, can use their own techniques and tools during the data modelling phase. None of the ASDMs specifies a specific data model (ERD or star-joins) that must be followed during the development of data warehouse, because every project has its own characteristics, resources, environmental circumstances and degree of uncertainty.

Due of the fact that Inmon (1996) does not collect requirements before data modelling or data staging takes place, it would be unwise to use ASDMs to try to develop a data model (star-join) based on his approach. Although there are some areas during data modelling and data staging where ASDMs could be applicable for Inmon's data warehouse, it will be of no value. The reason is that ASDMs emphasise that new or changed user and technical requirements must be incorporated in the design (using iterations) as development progresses, in order to give users what they want as fast as possible. Thus, no framework will be explained during this phase

Inmon uses ERDs and star-joins while Kimball uses a star-schema. However, the ASDMs do not explicitly state which data model must be used during the development of a data warehouse. In fact, most ASDMs do not even mention the use of a data model during development. They merely explain a process model to get the job done in a constantly changing environment were user satisfaction is their primary concern.

4.3.3 Data staging

Inmon also uses the ETL-process. During this process, data cleansing begins by extracting the required data from trusted sources, transforming it with programmes or tools, and loading the transformed data into the central data warehouse. Inmon's approach is data-driven, i.e. data is viewed as the most important factor.

The ETL-process can be replaced by the actual development phase(s) of every ASDM. The only difference between Kimball and Inmon's ETL-processes is that during Inmon's ETL-process new development requirements (not the requirements of the user) can be added after the loading process if analysis was unsuccessful.

After a program has been written to extract and transform the required data from the data warehouse, the data should be analysed (i.e. combined, merged, and redefined). If the result of the analysis does not satisfy the specified needs, it causes iteration with redefined and new technical and development requirements through which iterative development is promoted. This, however, may cause a delay where the programme used for transformation must be modifiable and adaptable because code will be changed due to the change in requirements. Another advantage is that the final report contains the results of many processed iterations. A decision should be made to determine whether the final report should be institutionalised.

Although Inmon (1996:96) promotes iterative data warehouse development, this approach has a disadvantage during transformation and integration (Inmon, 1996:116–120). This could be attributed to the fact that users are not part of any extraction, transformation, or loading process. This emphasises that ASDMs will not be applicable when using Inmon's approach. ASDMs state that users must be part of the development process or ETL-process of a data warehousing project to ensure that requirements are understood and developed as specified. As a consequence, user expectations are met. Another disadvantage of using the data staging of Inmon's approach is that during the first few times the desired data for extraction is determined, only part of the required data will be retrieved from the data warehouse, because requirements will only be identified during the "feedback loop". Consequently, time is wasted. It would therefore be unwise to develop a framework for the seven ASDMs during the data staging phase of Inmon's approach, as he does not include

users during the ETL-process.

4.3.4 Data access and deployment

Inmon (1996:128) defines two kinds of data access in a data warehouse: direct and indirect access. The data warehouse is deployed after alpha and beta testing has been completed and the data warehouse is populated from the existing system. After deployment, new requirements can be implemented using the "feedback loop" between the data architect and DSS analyst (see fig. 3.15).

According to Inmon (1996:66), there needs to be an "official organisational explanation (standards manual) and description of the data warehouse". Some aspects can be included in a manual in association with ASDMs that endeavour to minimise documentation. These are:

- A description of the source system feeding the data warehouse.
- How to use the data warehouse to gain maximum value (training).
- How to get help if there is a problem.
- Who is responsible for what?
- How data warehouse data relate to operational data.
- How to use data warehouse data for DSS.
- When not to add data to the data warehouse.
- What kind of data is not in the data warehouse?
- Guide to the available metadata.

Documentation of a data warehousing project using a specific ASDM can be in the form of a mini user manual that explains the user interface and effective use of the data warehouse. It could also take the form of a project plan, a star-schema (when Kimball's approach is used), or star-join (when Inmon's approach is used), the ERD tables, the source to target map, and data staging documentation on how data is cleansed, including quality issues.

Similar to Kimball's data warehouse, Inmon's data warehouse must be managed by creating backups and maintaining the data warehouse. In order for the data warehouse to be of value, the users must be trained to use the data warehouse effectively. The suitable and unsuitable characteristics of every ASDM will be explained during this phase because Inmon (1996:96) emphasises that "in all cases the data warehouse is best built iteratively", including iterative deployment. The main reason for paraphrasing the suitable and unsuitable characteristics is the unique feedback loop characteristic of Inmon's approach. This holds that newly identified requirements can be incorporated by the data architect into the existing data warehouse. Inmon's deployment differs from Kimball's deployment in that new requirements can be added to the existing data warehouse using the feedback loop. Another reason for building a framework for this phase is Inmon's (1996:66) preference for the use of a manual. One should take cognisance of the fact that because ASDMs are "lite" they keep documentation to a minimum. Par. 4.3.4.1 – par. 4.3.4.7 differ in many aspects to Kimball's approach using ASDMs to develop a data warehouse.

4.3.4.1 Dynamic System Development Methodology (DSDM)

Suitable characteristics

DSDM boasts the best supported documentation and training of any ASDM (Highsmith, 2002a:8). Like Inmon (1996:96), using DSDM, each increment, or iteration in the case of large projects, can be developed, tested and deployed during the development process. This concurs with Inmon (1996:96).

Incremental delivery and deployment is key to acceptable business solutions. The latter is one of the nine principles reflecting core values of agile processes.

In the "functional model iteration", every iteration is planned, reviewed, and then analyzed. Testing is integrated throughout the development process. The

"functional review documents", an output of the functional model, collects requirements from users from the current increments that can be used during other increments – just like Inmon's "feedback loop".

During "risk analysis" for further development, a document is created outlining risks, conclusions and new requirements, similar to Inmon's (1996:348) final report. (Risk analysis is another of the outputs of the functional model.) The output of the "design and build" phase is a tested system or data warehouse that meets at least the most important requirements set by users.

Implementation of a data warehousing project can occur during the implementation phase of DSDM, i.e. the deployment of the data warehouse within the user organisation. Output of deployment or implementation is a user manual that can train users to utilize the implemented data warehouse effectively.

The "incremental review document" can be used during the implementation phase to discuss the state of the system or data warehouse by testing and preparing it for further growth and maintenance. The "project review document" can be used to ascertain what to maintain in the developed data warehouse as well as what preparations should be made for possible growth.

During the "post-project" phase, the system is thoroughly tested and maintained to ensure that the deployed data warehouse continues to be of value and use.

Unsuitable characteristics

Inmon promotes the use of a manual (documentation). DSDM is "lightweight", i.e. it does not focus on documentation. There is also no definite way specified by DSDM of how data should be accessed or which reporting tools should be used.

4.3.4.2 Scrum

Suitable characteristics

As explained in par. 4.2.3.4, "planning" is a sub-phase of the pre-game phase that "includes the definition of the system being developed, project team, tools and other resources, risk assessment and controlling issues, training needs and verification management approval" (Abrahamsson *et al.*, 2002:29). Planning can thus be used to plan a training program for users.

The "architecture" phase, another pre-game sub-phase, includes the changes and the problems the changes may cause in implementing the product backlog if the implemented system (data warehouse) requires enhancement (Abrahamsson *et al.*, 2002:29).

Development is incremental. Each sprint include requirements, analysis, design, evaluation and deployment (Abrahamsson *et al.*, 2002:30). Inmon prefers this approach. During the post-sprint meeting it can be determined whether incrementally implemented design is suitable according to user expectations

One of the key principles of Scrum is that the system (data warehouse) must be constantly tested and documented.

Unsuitable characteristics

Scrum focuses on the development and success of development and therefore does not specify a specific way of implementing a data warehouse. The same applies to data access and reporting tools.

4.3.4.3 Extreme Programming (XP)

Suitable characteristics

Simplicity requires developers to keep the design as simple as possible so that

the necessary modifications can be implemented. Inmon's "feedback loop" can replace "feedback", one of the four values of XP where users define requirements on story cards.

"Testing" is a continuous activity throughout the development process of XP and data warehousing. In XP, "designing" is a continuous activity to incorporate new requirements into the evolving system – just like Inmon's "feedback loop".

XP delivers the system in small, tested iteration releases of business value to users. This approach is also favoured by Inmon (1996).

If the system (data warehouse) no longer holds value, changes will be made during "refactoring". Each time a task is completed, it is integrated into the system (data warehouse). Tests are then run, and should be passed for the changes in code to be accepted (Abrahamsson *et al.*, 2002:24; Cohen *et al.*, 2003:12; Lindstrom & Jeffries, 2004:48). Inmon promotes this practice of XP.

Because development is incremental and iterative, requirements are developed, tested, and implemented incrementally.

Unsuitable characteristics

No specific way, except incremental implementation in "small releases" (iterations), is specified by XP (see par. 2.4.3). Another disadvantage is contrary to Inmon's preference of a user manual, XP prefers minimal documentation.

There is no definite method of data access specified by XP, nor does it explain reporting tools to be used.

4.3.4.4 Feature Driven Development (FDD)

Suitable characteristics

"Progress reporting", an FDD best practice, involves reporting on all completed sections (increments). This can be replaced by the "final report" of Inmon's approach. FDD encapsulates best practices and incremental development to manage and monitor the development process (Abrahamsson *et al.*, 2002:47; Hislop *et al.*, 2002:175) and the data warehouse after deployment is complete.

During plan by feature, chief programmers lead small teams in the analysis, design, and development of new features. During design by feature and build by feature, the requirements are planned in more detail, built, tested, and integrated iteratively. Inmon also promotes data warehouse development and deployment in this way.

Unsuitable characteristics

FDD primarily focuses on the design and building phases of the development process and it does not need a specific process model to succeed (Abrahamsson *et al.*, 2002:47). No specific way of implementation is specified by FDD, nor does it offer guidelines as to how data should be accessed, or which reporting tools are preferred.

4.3.4.5 Crystal Clear (CC)

Suitable characteristics

Since development is incremental and iterative, implementation is done incrementally (if the data warehousing project is large) until the data warehouse has been fully deployed. Using "frequent delivery", working code are tested and implemented into the existing system (data warehouse).

"Osmotic communication", i.e. communication between team members and users, is seen as very important. It can, however, be replaced by Inmon's

"feedback loop".

Data warehouse users can be trained by development experts to gain easy and valuable access.

Furthermore, CC has a technical environment where testing and controlling tasks, e.g. creating backups and merging changes, are done during automated testing, configuration management and frequent deliveries, which is to its advantage

Unsuitable characteristics

By replacing written documentation with "face-to-face communication", a system can be delivered with reduced reliance on documentation. No specific way of implementing a system or data warehouse is explained by CC, nor does it offer specifications on data access and reporting tools.

4.3.4.6 Adaptive Software Development (ASD)

Suitable characteristics

In designing large and complex problems, a large volume of information (data warehouse) must be collected and analyzed, as Inmon suggests. "Final Q/A and release" (see fig. 2.12) can be replaced by deployment and data access in a data warehousing project. "Feedback" from users, one of the review practices of ASD, can in turn be replaced by Inmon's "feedback loop". The technical quality is reviewed during maintenance of a data warehouse, which adds to ASD's suitability.

During a data warehousing project, the team's performance is monitored as a standard review practice. The (data warehouse) project status can be reviewed through applying maintenance controls and tests on the existing system or data warehouse to identify whether the system is of value and performs satisfactory.

The "learning loop" (and feedback, as explained above), gained from repeated quality reviews in fig 2.12, using maintenance controls and tests to deploy a data warehouse of value in a data warehousing project, can be replaced by Inmon's "feedback loop".

ASD can make large projects, such as data warehousing projects, a success by managing change and deploying an up-to-date product.

Unsuitable characteristics

Managers are more concerned with dealing with collaboration and concurrency than with the details of designing, coding and testing. There is no definite way specified by ASD of how data should be accessed or which reporting tools should be used.

4.3.4.7 Lean Development (LD)

Suitable characteristics

During "amplify learning"; feedback to users can be replaced by Inmon's "feedback loop". During "delay of commitment", one way to keep options open during system (data warehouse) development is to implement the system for future capabilities and add-ons.

"Deliver fast", one of the seven principles of LD, entails creating value as quickly as possible by allowing no delays during testing, integration and deployment. Inmon prefers this approach when deploying a data warehouse.

LD focuses on the "waste elimination", e.g. duplicate data, by looking at the flow of value from request to implementation.

During "empowerment of the team", the whole team (including on-site and other users) can be trained to use the system (data warehouse) effectively.

To achieve "perceived integrity", the correct requirements must be tested and implemented after development to give users what they expect.

Poppendieck (2003:7) states that tests should be done early, often, exhaustfully, and an automated test suite should be delivered as part of the product (data warehouse).

A mature system serves users quickly, repeatedly and reliably (Poppendieck, 2003:3), and delivers a system (data warehouse) of value to users.

Unsuitable characteristics

There is no definite data access method specified by LD, nor does it shed light on which reporting tools should be used. "Delay commitment" and "decide as late" (principles of LD) can delay requirements collection and development.

4.4 Choice of data warehouse approach

During the evaluation investigation of ASDMs' suitability towards each phase of data warehouse development for Inmon and Kimball's lifecycles, it is clear that Kimball's approach is more suitable for developing data warehouses in chapter 5. The interpretive experiment in chapter 5 will determine the suitability of the seven ASDMs for data warehouse development. The main reasons for this approach is that ASDMs collect requirements before development of a data warehouse takes place, concurring with Kimball *et al.* (1998). ASDMs realize that every project has its own characteristics, resources, environmental circumstances and degree of uncertainty. Most ASDMs state that any tools or techniques can be used by experienced and well-trained developers to get the job done, covering the fact that a data modelling technique (star-schema) and tools or manually written programs can be used during data staging.

The idea of developing in increments can be applied in data warehousing

projects where every data mart or sub-data mart can be viewed as an increment that must be developed. Each increment can then be developed in iterations where new and changed requirements can be added as development progresses for a specific sub-data mart. The sub-data mart can then be delivered to the data mart (after passing certain tests); where after the data mart can be deployed as whole to become part of the existing data warehouse. Each data mart is developed and deployed separately.

The ETL-process for every data mart can be done in increments (for large data warehousing projects) i.e. the ETL –process can be divided in three increments (extract, transform, and load) where every ETL increment is done iteratively. In small data warehousing project the ETL-process can also be done iteratively and not incrementally.

Inmon (1996), on the other hand, defines requirements after the data warehouse has been deployed. This fact emphasizes that ASDMs cannot be applied to develop a data warehouse using Inmon's approach, as ASDMs focuses on collecting requirements before development, and continues the collection of requirements as a system or data warehouse evolves to satisfy user expectations. With this said, the study is limited to further explain the suitability of ASDMs during data modelling and data staging of Inmon's approach. The main reason for explaining the suitable and unsuitable characteristics of ASDMs during the data access and deployment phase of Inmon's approach is because of the unique "feedback loop" characteristic of Inmon (1996), where the new and alternative requirements are implemented by the data architect into the existing data marts or data warehouse (see fig. 3.15) after deployment takes place.

CHAPTER 5

APPLICATION OF ASDMs ON DATA WAREHOUSE DEVELOPMENT

5.1 Introduction

In chapter 4, the seven ASDMs were evaluated by investigating their theoretical suitability in each phase of data warehouse development of Inmon and Kimball's approaches. As discussed in chapter 4, it was clear that Kimball's approach has more potential to be successful using ASDMs to develop a data warehouse than that of Inmon. Since Kimball's approach focuses on collecting requirements before development takes place (as is the case with ASDMs), it was selected for the interpretive experiment reported in this study.

In chapter 5 the theoretical deductions made in chapter 4 will be interpretively tested. In this chapter it will be explained how the interpretive experiment was designed, how the data was collected, and how the data was analysed for the seven different development teams. Seven teams of equal strength developed a data warehouse using Kimball's data warehouse development lifecycle. Each team used their assigned ASDM to guide their activities during the data warehouse development process.

Furthermore, it will be explained how the interpretive experiment was conducted and executed. Lastly, the researcher will determine whether the data warehousing projects of the seven teams were successful.

5.2 Research design

The research plan is presented in this section. It includes what was done (i.e. providing the problem description); what was expected from the development teams; how the participant profile was compiled; and how the experiment was

executed.

5.2.1 Research plan

The purpose of the practical part of the study was to determine whether the theoretical deductions made in chapter 4 are applicable in a real world environment using an interpretive experiment.

The interpretive experiment was conducted using seven groups or teams to develop a data warehouse using their specifically assigned ASDM. The results of the interpretive experiment will be combined with the theoretical deductions from chapter 4 to present findings for the use of ASDMs in data warehouse development in chapter 6.

Seven data warehouses where developed using different ASDMs. The details of the data warehouses developed are given in the next section.

5.2.2 Data warehouse description

Each team had to develop a medium to large cricket data warehouse using their randomly assigned ASDM. A number of specific requirements were expected from each team during the development of the data warehouse. The data warehouse users in this study were cricket supporters and team selectors.

The data used for the data warehouse was extracted from www.cricinfo.com. After successful extraction, the text files were processed into ERD tables and stored in a temporary MySql database, viewed as the source system for further processing. The data had to be extracted from the source system, transformed and cleansed before it could be loaded into Oracle. As part of the transformation process, an application was written to clean the data in order to load the cleansed data into the Oracle data mart (star-schema). Before the data was loaded, the participants in each group had to develop a star-schema (fact

table with facts and its corresponding dimension tables) in Oracle. The data warehouse only consisted of one data mart. Surrogate keys and key maps were used with the transaction ERD table (namely a Balls table) to populate the Oracle fact table.

After a successful loading operation, an application had to be developed that had to connect successfully to the Oracle constructed data mart. The application had to generate scorecards for cricket matches played, including five-day tests and ODI (one-day international) matches. Furthermore, the application had to be user friendly with a user-friendly query builder that constructs editable queries that would help users understand and make choices based on the statistics and information they receive from the executed queries. The fact table and its corresponding dimension tables had to be viewed in the application, implying that an effective connection had to be created between the application and data tables in Oracle.

Lastly, the teams had to use Cognos (a reporting tool) to generate reports from the data warehouse through an effective connection. After completing the project, the participants in each team explained whether their assigned ASDM was successful in developing a data warehouse by exploring the suitability of their assigned ASDM's properties, principles, practices or phases.

These were the primary requirements applicable to all ASDM teams.

5.2.3 Participant profile

The participants in the seven teams participating in the interpretive experiment were postgraduate students at a university. During the four years of study they learned to program in several different languages, including Java, Visual Basic, VB.NET and C#. Other modules such as project management, information systems development (including methodologies) and data warehousing were also part of their curriculum. They also completed four Oracle modules in two

years as part of their curriculum. All participants passed these modules, making them suitable for delivering a data warehouse consisting of one data mart to fulfil the requirements explained in par 5.2.2. This project formed part of a module of the BSc (Hons) programme. The participants were well equipped to complete the task of developing a data warehousing using a specifically assigned ASDM.

The participants were divided into groups, ensuring teams of equal strength. Development was done using groups or teams because of the heavy workload involved in developing a data warehouse. Most importantly, the reason for developing in teams, was that all ASDMs depended on teamwork during project development. This ensured thorough testing of the ASDMs. Teamwork is a common practice used in the development of large projects.

An ASDM was randomly assigned to each team to develop a data warehouse. The teams firstly had to study the ASDM assigned to them and present an approach in the first interview on how the assigned ASDM could be applied to develop a data warehouse using Kimball's approach.

5.2.4 Interpretive experiment description

The interpretive experience started by dividing the 19 participants into 7 groups or teams to develop a data warehouse using a randomly selected ASDM assigned to each team. The participants had to study their ASDM as well as the other ASDMs to get a solid background. The researcher used cross-case analysis (explained in par. 5.4) to analyse the data collected from the interviews, documentation and evaluation sessions.

Seven interviews comprising open-ended questions were conducted and recorded on a weekly basis. Each team firstly had to develop a project plan including a diagram and a schedule. The diagram had to bring the assigned ASDM and data warehousing together, showing how a data warehouse can be

developed, one increment (data mart or a part of a data mart) at a time, using the team's assigned ASDM (see par. 5.4). The requirements were given to all participants during a requirements session after the first interview, as explained under the second interview in par. 5.3.1.

Halfway through the project the data warehouse was evaluated during an evaluation session to determine whether the requirements set to date had been fulfilled. A second evaluation session was conducted after the data warehousing projects were completed to establish whether all primary requirements were fulfilled, and whether the projects were completed successfully. For the final evaluation session, the teams had to deliver primary documentation as described in par. 5.3.2.

5.3 Data collection

Data were collected in three different ways. The researcher conducted seven semi-structured interviews (open-ended questions) to understand the problems the team may have experienced. Primary documentation was completed, i.e. every team had to deliver certain important documents as specified in par. 5.3.2. Two evaluation sessions were held to determine whether the primary requirements given during the requirements session (explained in interview two) had been met.

To establish whether the data warehouse teams were successful, the following aspects were investigated:

- Did the team understand the ASDM assigned to them based on the questions asked during the first interview?
- Were the ERD and star-schema constructed correctly?
- Was the source-to-target map done correctly?
- Was the fact table loaded correctly using key maps and look-ups?
- Was data staging done correctly (i.e. created and deleted data are

acceptable and data were cleansed correctly)?

- Was the GUI user friendly and was an effective connection created with the data warehouse?
- Did the GUI have an editable query builder?
- Was the data warehouse effective?

5.3.1 Interviews

Weekly interviews were conducted with all participants of every team. The last interview was held five days before the final evaluation session and four weeks after the sixth interview. This four-week interval was necessary for the participants to finish their data warehouses. The seven 15-20 minute interviews were recorded on a laptop.

Interview 1

In this interview the attitudes and perceptions of the team members were tested after the ASDMs were assigned to each team (see par. 5.2.4). The researcher wanted to determine whether the team comprehends where their assigned ASDM can be applied in the data warehouse development lifecycle. Furthermore, a project leader was assigned for every group and the participants were asked to come prepared for the requirements session held a day before the second interview.

Each team was asked the following questions to determine whether they understood their assigned ASDM, and the other ASDMs:

- What is an ASDM?
- Did the team study all seven ASDMs?
- Explain the assigned ASDM.
- After studying their assigned ASDM, does the team think it has the potential to be used to develop a data warehouse?
- Does the team think that development can be done in increments and

iterations?

- Why would it be a good idea to develop in increments and iterations?
- How does the team feel about the project?
- Does the team have any questions?

The teams were then asked to develop a project plan and schedule, including a diagram, explaining how they think their ASDM fitted into the development of a data warehouse. The first project plan was submitted before the second interview.

Interview 2

At this stage the requirements given during the requirements session were discussed. The users wanted to examine the same kind of information given during a cricket match on television, including information they could interpret to decide which player to choose for a certain match, based on the player's statistical information. These statistics included, for example, against whom (which country) and where (in which country) did a certain player perform better. The primary user requirements provided during the requirements session were the following:

1. The most important user requirement was that the teams had to generate a scorecard for every match. The scorecard had to include the following:
   - Name of each team in the specified match.

   Batsman:

   - Names of batsman.
   - Score of batsman.
   - Number of overs (balls) bowled to batsman.
   - Number of fours and sixes hit by a specific batsman during the match.
   - If a batsman was dismissed, the method of dismissal and which bowler was responsible.

Bowler:

- Name of bowler.
- Total number of overs bowled by a specific bowler.
- Other technical information such as wides and no balls bowled.

The GUI had to be user friendly. The participants in each team were responsible to add their own technical requirements, and to meet these requirements. To generate a scorecard for users will require a very large query.

2. Additional user requirements were given based on individual player performances, including:
   - The players' batting average.
   - The most runs a player accumulated during an ODI and five-day test.
   - How many five-day tests and ODI matches has the batsman played?
   - How many 50's and 100's (centuries) did the batsman score?

   If the corresponding player was a bowler, the following requirements should have been provided:
   - What was the average number of runs a batsman hit from a bowler before being dismissed (bowling average)?
   - What were the biggest number of overs (runs) bowled in a specific match?
   - How many hattricks (three people taken out with three consecutive balls) has the bowler had in his career?
3. User requirements that would help a selector choose specific players for a match based on a player's statistics:
   - How many times have two teams played against each other, in which years, and who won? What was the score during every inning if it was a five-day test?
   - Which batsman hit the most runs in an ODI and five-day test match and against whom – including his personal statistics?
   - Which bowler took the most wickets from a specific team?
   - How many times did a specific bowler take a wicket after the batsman hit

four runs?

- How long did a specific batsman continue batting after scoring 50 runs?

During the second interview, the researcher investigated whether the teams understood the requirements given to them during the requirements session.

The project plan (including a diagram, ERD tables, star-schema and schedule) that was submitted five days prior to the interview, was evaluated by the researcher. Each team's diagram was discussed, establishing whether the participants understood how to apply their ASDM to data warehouse development.

Interview 3

During this interview the second project plan was evaluated and reviewed. New requirements were added, including designing a source-to-target map indicating where the star-schema's data came from. The final project plan had to be submitted four days before the fourth interview. Each team was expected to adhere to a task time schedule. The researcher reviewed whether each team was on schedule on a weekly basis. All the teams were placed on the same schedule without changing the characteristics of each project, e.g. every team had to finish certain subjects of the project before a certain date. Each team still used their own cycles, story cards, features, sprints or iterations to meet these deadlines. This helped the researcher to manage all the projects by stating, for example, that all teams' fact table had to be loaded before a certain date.

Interview 4

The respective team project plans, ERDs and star-schemas were reviewed to determine whether the teams were working according to their schedules. Last chances were given in cases where the researcher identified problems. The researcher allowed team characteristics to show in the data warehousing

project and development process. During this interview, the researcher stated that all dimensions for each team should be loaded before the next interview. The following aspects where investigated:

- Did the team understand why a project plan was developed? The project plan brought the team's thoughts in line with what should have been done next to keep the team on schedule, so the project could be completed in time. Furthermore, the diagram helped the team understand how the ASDM could be applied to develop their data warehouse.
- Did the team have any team problems?
  All teams were working together effectively without any major problems.

Interview 5

The purpose of this interview was to discuss problems within the team or the development process. Every team's dimensions had to be loaded successfully. Every team's fact table should have been loaded before the next interview, to ensure that the data warehouse would be delivered on time.

Interview 6

During this interview, the schedule was discussed. New requirements were added for the first evaluation session (explained in par 5.3.2) where each team had to have data staging documentation indicating how the data were cleansed.

Interview 7

The purpose of this interview was to understand all the problems the teams experienced in the final stages of their data warehouse development. Five of the seven groups where not sure how to implement user requirements in Cognos. The users were contacted to provide technical detail to these students.

5.3.2 Project documentation

ASDM does not focus on documentation. It was decided that only the most important documentation involved in developing a data warehouse should be produced, and not all the documentation as specified by Kimball *et al.* (1998).

Firstly, the team had to develop a project plan with a diagram, ERD, star-schema and schedule. The diagram had to explain how the assigned ASDM could be applied to the data warehouse development lifecycle of Kimball's approach. The researcher could also identify in the diagram whether the team applied the assigned ASDM correctly to data warehouse development. Every diagram had to show that development takes place in increments (sub-data mart, or data mart in the case of CC in par. 5.4.5) iterations. Every increment or sub-data mart was developed iteratively and delivered incrementally to create the data mart, where after the data mart (including everything from collecting requirements to GUI development and report generation) was deployed as a whole to the users.

The diagrams developed indicated that iterative development is suitable where a logical grouping of tasks (iteration) were completed successfully before starting the next logical grouping of tasks. After one increment (data mart or a sub-data mart) was fully designed, the next increment could begin by for example developing the GUI, or extracting and collecting the requirements for the next data mart that could also be developed in iterations. The participants only needed to develop one data mart with a grain of ball for ball data in cricket match history.

The project plan was very important because it helped the teams to get their thoughts in order regarding what was expected from the data warehouse, and how their assigned ASDM could be applied to data warehouse development (explained by the diagram). It also helped to manage the participants' time through the use of a schedule. A star-schema using Kimball's approach had to

be designed from the ERD tables in the database. Furthermore, additional requirements were added, e.g. the development of a source-to-target map, keeping the project in an agile environment. New data staging and quality issue documentation (requirements) were also added during development. Data staging documentation was submitted for the first evaluation session, showing how the data was cleansed and loaded into the data warehouse, and using screenshots of the cleansing application as part of the explanation. The quality issue documentation explained the problems with the data, data that was generated and deleted. This document was important because it ensured the researcher could identify whether the data was cleansed correctly, making sure for example that no important data was deleted, or that data with no value was generated.

In addition, the teams had to explain how they populated their fact tables. This was very important because if the fact table was incorrect, the whole data warehouse would be of no value.

As part of the final documentation that had to be completed for the last evaluation session, the participants had to produce a mini-manual (keeping the environment agile). A large manual was not required, because ASDM does not focus on documentation (Lindvall *et al.*, 2002:206). Because seven different data warehouses and GUI applications were developed, the mini-manual had to explain (using screen shots) how a user could use the data warehouse and GUI effectively. The mini-manual also had to include how Cognos was used to generate a report (with screenshots). This was necessary to ensure that the user used all aspects of the application, and to determine whether all the primary requirements were fulfilled.

The teams had to submit a written reflection on their teamwork as well as on their development process for the final evaluation session. They had to explain whether the assigned ASDM worked for the development of the data warehouse. The team had to further explain in which areas, properties or

principles the assigned ASDM worked during the development of a data warehouse, including advantages and disadvantages of their assigned ASDM, where possible. The team's explanation had to include a reflection on the problems they experienced as a group during data warehouse development, and whether the assigned ASDM helped them to solve some of these problems.

Finally, the effectiveness of the data warehouse had to be tested. For this purpose, the team had to execute a complex query on the database containing the source ERD tables, and the data warehouse containing the fact table and its corresponding dimension tables. As part of the final documentation, the SQL statement used needed to be explained for both the database and data warehouse. This was very important, as the data warehouse had to be effective to be of any value.

Because documentation was kept to a minimum, the participants could focus on satisfying the user requirements, making these data warehouse projects "lean".

5.3.3 Evaluation sessions

There were only two evaluation sessions during the development of the data warehouses, during which each team had to give a presentation of their data warehouse's progress. An explanation of what was evaluated and what should be presented during each evaluation session follows.

Evaluation session 1

The evaluation session took place seven days after the sixth interview. An e-mail was sent two days before this evaluation session including the evaluation sheet (see Table 5.1) and presentation requirements. Each team had approximately 35 minutes to present their data warehouse. The user

requirements for the presentation and evaluation session included the following:

- Data staging documentation, showing how the data was cleansed.
- ERD and star-schema.
- The GUI should have had an effective connection with the Oracle data mart.
- The users should have been able to view all dimension tables and fact tables in the GUI.
- The applications should have had a primitive query builder where queries could be executed on the data mart.
- The GUI should have been understandable and easy to use.
- The team should have indicated how the fact table and facts were loaded, using the GUI.
- The team should have worked out three queries that would be of value to the users.
- The assigned ASDM had to be applied correctly during data warehouse development.

The following evaluation sheet shows the different aspects evaluated in the data warehouse evaluation session for each team:

Subject evaluated
ERD
Star-schema
Load of dimensions
Load of fact table
Load of facts
Quality issue explanation and data staging
GUI
Queries
Teamwork
Use of ASDM
Total

Table 5.1: First evaluation sheet

Evaluation session 2 (Final evaluation session)

The final evaluation session took place three days after the seventh interview. During this session, each team evaluated and presented the final data warehouse project. The documentation requirements and evaluation sheet (see Table 5.2) was e-mailed five days prior to the final evaluation session (keeping the environment agile). The documentation was written in a specifically organised format with the following headings:

- Project plan: Updated project plan and schedule.
- ERD: Updated ERD diagram.
- Star-schema: Updated star-schema.
- Data staging and quality issues: Updated data staging documentation and data quality issue documentation (data problems, generated and deleted data), including screenshots.
- Load of fact table and facts: Explanation of how the fact table and facts were loaded.
- Mini-manual: Explaining the effective use of the data warehouse, including a Cognos explanation, both with screenshots.
- Evaluation of assigned ASDM in data warehouse development: Explaining how the assigned ASDM was used to develop the data warehouse, including which phases, properties, practices or principles worked for data warehouse development. Advantages and disadvantages could also be included.
- Problems during data warehouse development: What were the problems the team encountered during the development of the data warehouse?
- Data warehouse effectiveness: Present a large query to the database and the Oracle data warehouse to determine whether the data warehouse was effective and accurate. The data warehouse will be effective if the same answer or a maximum error margin of 5% was given.

Each team was granted only 20 minutes to present their data warehouse. The rest of the time was set aside for the users to ask questions. Each team was

granted an evaluation time of approximately 50 minutes.

Subject evaluation
Section 1: Presentation of data
SQL queries
Connection between application and Oracle
GUI and generated queries
Scorecards
Cognos
Section 2: Business intelligence
Are queries of value to the user?
Presentation of results
Section 3: Teamwork and use of assigned ASDM
Effectiveness of group
ASDM application
Data warehouse effectiveness
Total

Table 5.2: Final evaluation sheet

The evaluation session included all three sections as seen in table 5.2. During section one, the users wanted to evaluate the SQL queries generated by the participants of a specific group. The team had to show an effective connection between the Visual Basic or C# GUI and data warehouse. The GUI and query builder had to be easy to use, generating a scorecard for any specific match. Cognos had to be presented, showing at least one cube and a drill-down operation. A report also had to be generated and presented in Cognos that would be of value to the users.

In section two, the evaluators determined whether the generated queries were of value to the users. Furthermore, the results of any selected or built query had to be presented in an understandable format.

In section three, the effectiveness of the team members were evaluated to see whether they were team players. The team had to explain whether their ASDM was applicable during data warehouse development. They had to state whether there were any limitations to the assigned ASDM. As in evaluation session one, the team had to explain in which phases, properties, practices or principles their ASDM worked or did not work. They also had to recommend how the ASDM could be adjusted to make it more effective. Finally, the effectiveness of the data warehouse was tested by executing a complex query on both the database and data warehouse.

The teams were evaluated from the end users' perspective. For example, if one team's query builder was easier to use than another team's query builder, they received more evaluation points.

5.4 Data analysis

A large amount of data was collected during the development of the seven data warehouses, including seven interviews, two evaluation sessions, three updated project plans, first evaluation documentation and final evaluation documentation. The cricket source data used by the participants to develop their data warehouse came from a real world source and data inconsistency problems cannot be associated with the academic setting of this interpretive experiment where ASDMs were used to develop data warehouses. This study focused on data directly applicable to the application of ASDMs on data warehousing.

The researcher used cross-case analysis as described by Seaman (1999:568) to analyse the findings of the interpretive experiment. This is done by firstly analysing the data collected from the first two ASDMs (DSDM and Scrum); by writing descriptions in a list that describes each case evaluated. The two lists were then compared to determine the similarities, which were supporting

evidences of the suitability of ASDMs in data warehouse development, and differences. The next step was to list propositions and their supporting evidences if the two ASDMs were the only ones analyzed. After analyzing the first two ASDMs, the third ASDM (XP) was examined and a list of XP's characteristics was compiled. It was then determined whether the third ASDM supported any of the propositions formulated from the first two ASDMs, namely, DSDM and Scrum. If a proposition was supported, the third ASDM was added to the list of supporting evidence. If the third ASDM (XP) contradicted a proposition, then either the proposition was modified or the description was noted as refuting that proposition. After this was done, any additional propositions were added to the list. This process was repeated with each ASDM case evaluated.

Propositions resulting from cross-case analysis

As a result, the following list of propositions (rich in detail), was determined:

- ASDMs do not prescribe a specific structure (structure can change depending on the type of project) of how a project should be developed (DSDM, Scrum, XP, CC, ASD, and LD).
- The developed project plan and schedule supported the team to get their thoughts and ideas in order (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- Active user involvement when applying ASDMs on data warehouse development is imperative (DSDM, Scrum, XP, CC, ASD, and LD).
- Using the assigned ASDM, data marts can be delivered incrementally (DSDM, Scrum, XP, FDD, ASD, and LD). Each team proved this by means of the diagram develop for their assigned ASDM.
- Using the assigned ASDM, a sub-data mart can be developed in iterations, sprints, cycles, feature sets and sub-sets depending on the ASDM (DSDM, Scrum, XP, FDD, ASD, and LD).
- Using the assigned ASDMs a data mart (including everything from collecting requirements to GUI development and report generation) can be deployed as a whole to the users (DSDM, Scrum, XP, FDD, ASD, and LD).

- When applying ASDMs to data warehouse development, the design is kept simple (XP and LD).
- The sub-data mart logical structure (including everything from collecting requirements to GUI and report creation) was easily divided into individual tasks, iterations, features, sprints or cycles (DSDM, Scrum, XP, FDD,, ASD, and LD).
- Because data warehouse development was done incrementally and iteratively, problems were more easily identified and solved in less time (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- The data warehouse design improved as incremental and iterative development progressed (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- Development is revisable when ASDMs were applied to data warehouse development where participants could backtrack (using spike solutions or post-sprint meetings to identify solutions) to the last safe point in order to start a new development approach (DSDM, Scrum, XP, ASD, and LD).
- Regular meetings (pre- and post-sprint meetings) helped teams to understand what was expected of them during each development iteration (Scrum, XP, FDD, CC, ASD, and LD).
- The ASDM team should be empowered to make their own decisions (DSDM, Scrum, XP, ASD, and LD)
- ASDMs emphasise teamwork when a data warehouse is developed (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- ASDMs allow the team to learn as development progresses, making development a learning activity (Scrum, XP, CC, ASD, and LD).
- Teams struggled to stay on schedule, because the users kept the environment agile by adding and changing requirements (DSDM, Scrum, FDD, CC, ASD, and LD).
- The ASDMs encourages and improves development of a data warehouse in an agile environment created by the users (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- ASDMs encourage feedback as development progresses (Scrum, XP, and

LD).

- Communication between team members, users and other stakeholders is very important when developing a data warehouse (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- If the team is not committed to the data warehousing project, the process of successful completion could collapse (Scrum, XP, FDD, CC, and LD).
- ASDMs allow developers to use tools and techniques from other methodologies, for example, where pair programming was used by all the ASDMs except DSDM (Scrum, XP, FDD, CC, ASD, and LD).
- Access to expert users when developing a data warehouse using a specified ASDM, is of the utmost importance (DSDM, Scrum, XP, CC, ASD, and LD).
- Frequent delivery (in feature sets, sprints or cycles) and constant integration is essential to project success (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- Testing can be integrated throughout data warehouse development when applying a specific ASDM (DSDM, Scrum, XP, ASD, and LD).
- There was a lack of user involvement in the data warehousing project (XP, Scrum, and LD).
- New and additional requirements were implemented without any major problems (DSDM, Scrum, XP, FDD, CC, ASD, and LD).
- ASDMs do not have capabilities to solve technical issues during data warehouse development (DSDM, Scrum, FDD, CC, ASD, and LD).
- ASDMs will be more applicable to larger projects than the medium-sized data warehouse developed by the teams (DSDM, FDD, and LD).
- ASDMs have the ability to be used in data warehouse development (DSDM, Scrum, XP, FDD, CC, ASD, and LD).

There is a definite correlation between the core values of agile processes, as explained by Hislop *et al.* (2002:177) (see par. 2.3), and the propositions derived by the researcher during the interpretive experiment. All the core values specified for agile processes were found to be true for all ASDMs used in this

interpretive study.

Propositions for individual ASDMs

The unique characteristics of each ASDM will be discussed in terms of:

- *Data warehouse diagram*: The diagram depicts how the team's assigned ASDM was applied to the development of their data warehouse. The diagram describes an incremental data mart development approach where development is done one sub-data mart (increment) at a time. The development of a data mart includes all the increments associated with the data mart, i.e. the diagram of the collected requirements into a star-schema; the design and data staging of the star-schema; the GUI associated with the data mart, and the reports generated according to user specifications. Each increment was developed iteratively to deliver the increments (sub-data marts) to create the data mart, where after the data mart was deployed as a whole. In this interpretive experiment, the cricket data warehouse consisted of only one data mart that was divided in increments (sub-data marts), each developed iteratively. Another data mart may for example be a financial contract data mart where players' contracts are managed based on performance. The CC team was the only team that viewed an increment as a data mart, where the whole data mart was developed iteratively and deployed as a whole.
- *The team's interpretation of the assigned ASDM's suitability towards data warehouse development before development*: The assigned ASDM will be evaluated by the answers to the following question in the first interview: "After studying their assigned ASDM, does the team think it has the potential to be used to develop a data warehouse?"
- *The team's interpretation of the assigned ASDM's suitability towards data warehouse development after development*: The assigned ASDM will be evaluated by the answers to the questions in the final evaluation session: "Did the assigned ASDM work for developing the data warehouse? In which areas, properties or principles (depending on characteristics of ASDM) did

the methodology work and in which areas, properties or principles did it not work? Are there any shortcomings in the assigned ASDM?"

- *Documentation evaluation:* The assigned ASDM will be evaluated by examining the documentation provided for the final evaluation session.
- *Problems during development:* Problems encountered during data warehouse development will be explained, including whether the assigned ASDM helped to manage the problems.

5.4.1 DSDM team

Data warehouse diagram:

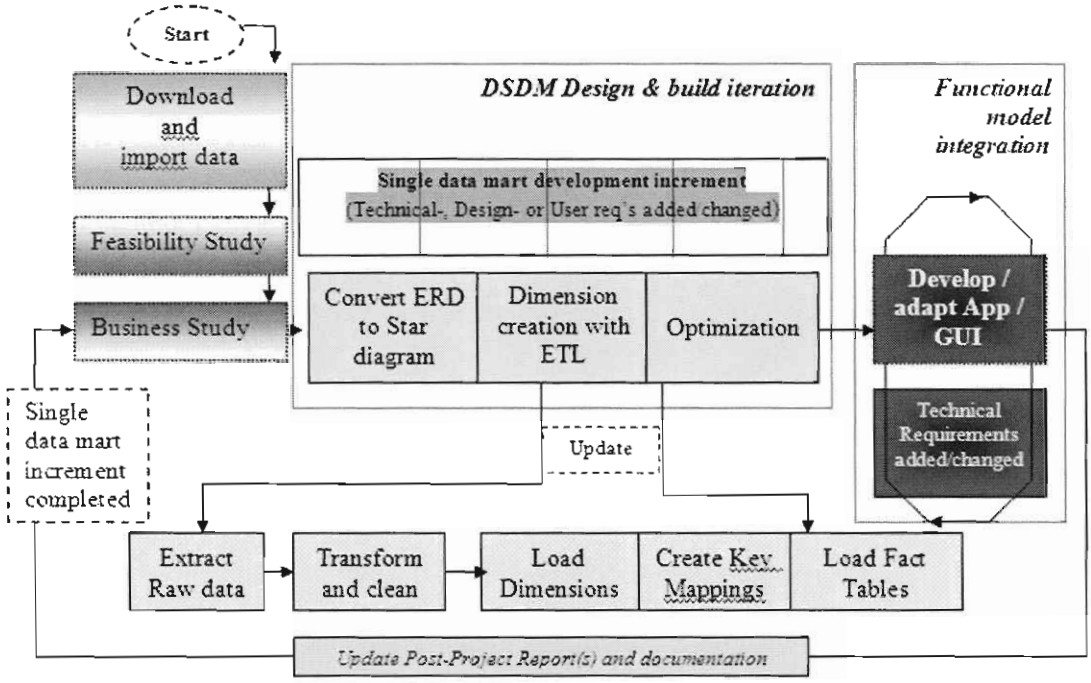


Figure 5.1: Applying DSDM to data warehouse development.

The diagram depicts definite development phases or studies that distinguishes DSDM from other ASDMs. In each phase certain aspects of data warehousing could be incorporated, for example the conversion of ERD tables to a star-schema could easily be incorporated as an iteration as part of the design and

build phase of DSDM. Furthermore, the design and build iteration is clearly visible on the diagram.

The team's interpretation of DSDM's suitability towards data warehouse development before development:

The participants replied: "If we know what the user wants, we will know if the DSDM will work." This implies that the team members were uncertain whether DSDM would work without having knowledge of the primary requirements.

The team's interpretation of DSDM's suitability towards data warehouse development after development:

Because the users kept the environment agile, the DSDM team experienced a problem because DSDM states that time should be fixed while functionality is adjusted. The deadline was set by the users, but they underestimated the duration of their individual tasks, causing the team to change the timeframe of individual tasks to ensure that the functionality stayed the same. As new requirements were added, the functionality declined – contrary to the key aspect of DSDM.

Documentation evaluation:

DSDM has proven to extend the duration of individual tasks in the project timeframe. This is in contrast with the statement of DSDM implementation specifications, which explain that the project timeframe should be fixed. Functionality was flexible and if changed, it was supposed to decrease in order for the project to meet the time constraints. In this data warehousing project, however, functionality requirements kept on increasing, compromising time constraints.

The DSDM team do, however, state that DSDM can be used for data

warehouse development if the functionality requirements are set (which is rarely the case in any software project). According to the team, user involvement is important because users do not know exactly what they want at the beginning of a project. They explained that incremental data mart development worked throughout the development phases of the DSDM's lifecycle.

Problems during development

Apart from the technical issues experienced by the team, the team struggled to divide the workload among the team members, as the development experience of each team member was not known.

5.4.2 Scrum team

Data warehouse diagram:

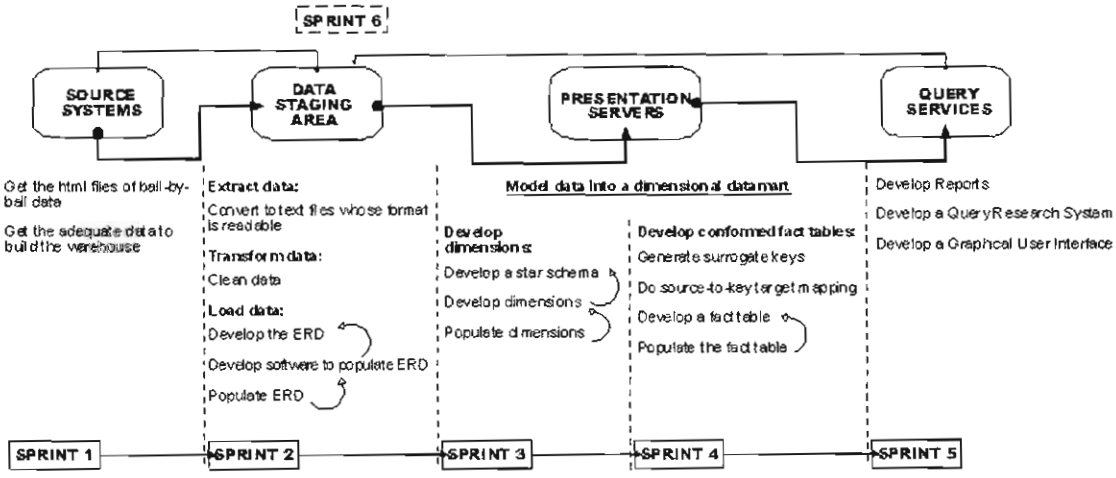


Figure 5.2: Applying Scrum to data warehouse development

Scrum's development process is uniquely identified by the sprints used during the data warehousing project. Pre-sprint meetings where held to ensure that all team members recognise what was expected from the current sprint. Each sprint represented a logical grouping of tasks that had to be completed before

the next sprint could commence. After each sprint, a post-sprint meeting was held to determine whether the requirements from the sprint where met and whether there were any additional identified requirements to be included in the next sprint.

The Scrum team implemented Scrum in seven sprints in the project. During sprint three (data modelling), they had several ideas among them and were able to implement most of these through programming and developing together using individual strengths. Some of them had more knowledge about cricket and others had more knowledge about the programming procedures.

During sprint four (population of fact table) the fact table was loaded within three hours, meaning that Scrum had no impact on this sprint as a normal sprint period is 20 days or less.

During sprint five (creation of GUI) Scrum was supportive. The team had several meetings and through working closely together they knew who would be best for which part of the GUI.

During sprint seven (project report) Scrum was a huge help. Due to all the prior notes and reports it was easy to update schedules and get required documentation ready through previously developed documentation in the other sprints.

The first two sprints (methodology and ERD development) did not include any development process. For this reason Scrum was not used, while in sprint six (building a cube in Cognos) the team only used the pre- and post-sprint meetings as specified by Scrum. According to the Scrum team, Scrum was very effective in developing their data warehouse.

***The team's interpretation of Scrum's suitability towards data warehouse development before development*:**

The participants gave a positive response to this question but did not give supporting detail.

***The team's interpretation of Scrum's suitability towards data warehouse development after development*:**

The Scrum team stated that the daily meetings improved communication, causing each participant to know exactly what was expected during the current and future sprints. The team explained that, "...because the work was divided in sprints, individual tasks of the project could be completed easier". The team saw a sprint as a logical grouping of tasks to be completed before the next logical grouping of tasks could start.

Documentation evaluation:

The Scrum team explained some unique advantages and disadvantages they experienced during the development of their data warehouse.

Advantages:

- Rotation of leadership depending on the development phase gave a distributed nature of project execution and ownership. Every member had the opportunity to be a sprint leader. In this way not only one person's development characteristics were imprinted upon the project.
- The users were kept informed of the progress of the team and it was possible to step in whenever required. This was greatly extended by the weekly reports as well as the meticulous notes the team kept of every meeting.
- Scrum creates an open environment and encourages feedback.
- Evaluation of effort and subsequent rewards were based on team performance.

- Reduced need for meetings, authorisation and reporting. As they grew to know each other's working style, the team held shorter meetings and worked much faster.
- The incremental data warehouse development model allowed the team to deliver every 20 days or less, and not the normal 30 days as specified by Scrum.

Disadvantages:

- Loss of initiative and great ideas: Ideas the team had could not be implemented since, according to them, they did not have enough time because of the changing environment.
- Emotional impact when not keeping to the schedule: The team members feel they are working continuously and are constantly behind schedule with the "malicious hastening" of the project.
- As with other ASDMs, the Scrum team experienced some technical and source data problems.

Problems during development

The Scrum team experienced the following categorised problems:

- *Problems with the data warehouse:* The team argue they wanted to add more functions and tools to their data warehouse, but it was difficult with the additional requirements they had to complete.
- *Problems with users:* The Scrum team argue that they struggled with the users who constantly changed the requirements. They further argue that they received some of the assignments at too late a stage, causing them to not implement the solutions to the best of their abilities. (The researcher added requirements on purpose to keep the development environment agile.) The team suggests that less time be spent on the initial planning. Because of the constant changes, data became redundant and caused many inconsistent dates and tasks. The team had to spend several hours

to correct this problem near the end.

- *Problems among the team members:* The Scrum team explains that they were fortunate to have a group with such diverse strengths. The team did not experience any problems that could not be sorted out quickly and efficiently.

5.4.3 XP team

Data warehouse diagram:

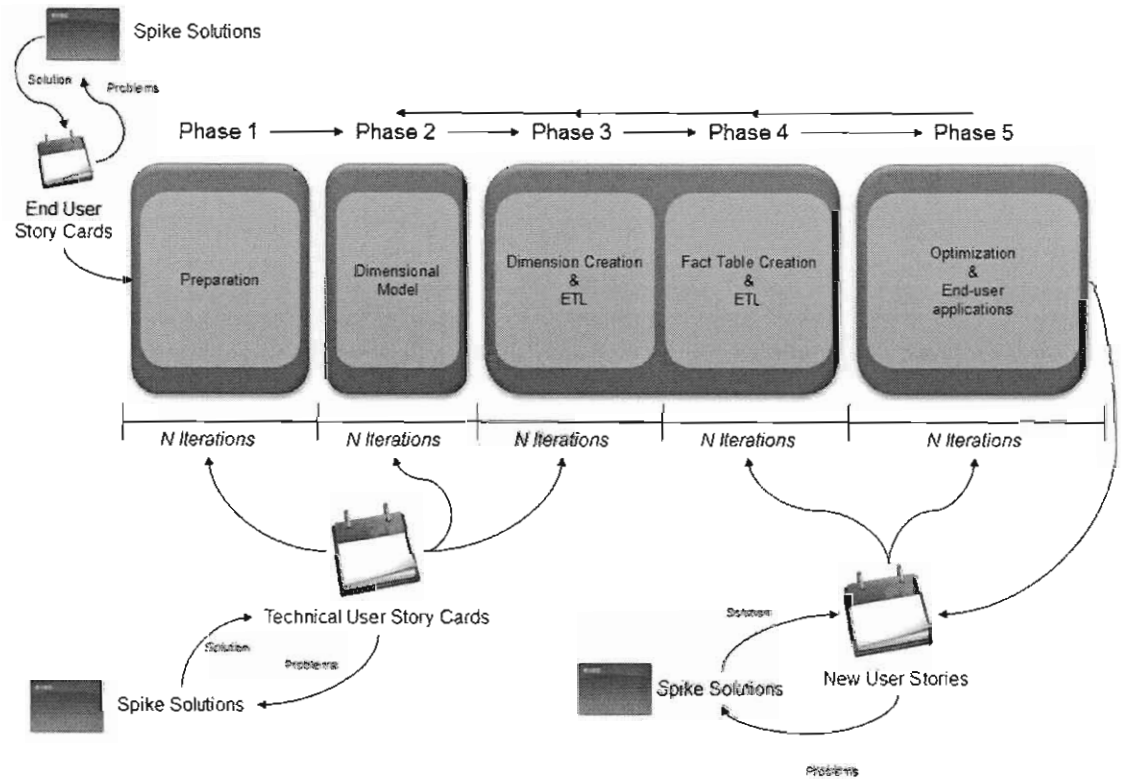


Figure 5.3: Applying XP to data warehouse development

The story cards that were used to identify requirements and to guide the development process uniquely identify XP. After the completion of a logical grouping of tasks or requirements on a story card, spike solutions were used to identify problems experienced during iterative development to find and incorporate appropriate solutions.

***The team's interpretation of XP's suitability towards data warehouse development before development*:**

The participants gave a positive response, but explained that they were not sure exactly what was expected from them. This was resolved during the requirements session.

***The team's interpretation of XP's suitability towards data warehouse development after development*:**

The XP team was extremely positive towards the development of their data warehouse using XP and replied: "...we think that we had the best ASDM. XP worked very nicely for us. We completed story cards ourselves as development took place and as we realized that new requirements must be added to make the data warehouse more valuable. Technical requirements, which were our responsibility, were also filled onto story cards to get our thoughts and ideas aligned to what was expected. We used pair programming the whole time to program and design. Pair programming helped us to fix errors much faster, as the one programmer may know more than the other, creating a learning environment for us as team members. We think that XP also has the ability to be used in conjunction with other ASDMs."

Documentation evaluation:

According to the XP team, XP was a sufficient methodology for developing the data warehouse.

The most effective method of evaluating the failures and successes of XP was to evaluate the core practices of XP in terms of the data warehouse project. All XP practices (continuous integration, design improvement, small releases, simple design, etc.) were applicable in the use of other ASDMs, as explained in

par. 5.4.

The following practices were unique to XP's character during data warehouse development:

1. *Test driven development customer tests and on-site customers:* The only problem with the simulated environment was the lack of consistent everyday user involvement. Acting as both users and developers made objective development quite difficult, reason being that XP was built upon user involvement and thus testing and quality assurance are user dependent. According to the team, no iteration can be completed without user acceptance after extensive testing.
2. *Planning game (story cards):* The simulated environment made it quite difficult to fully practice the planning game. However, the XP team do think that it is applicable to data warehousing because the story cards kept the requirements short and focused.
3. *Coding standards or coding conventions:* Using XP's recommended coding standards and conventions was a viable idea with the focus on metadata.

In conclusion, the XP team found XP to be a methodology that is applicable to the data warehousing paradigm overall. Except for the problems with the simulated environment, they would definitely use this methodology for future data warehousing development.

Problems during development

Although XP has nothing to do with the technical issues, the XP team explains that XP helped with technical problems by offering the option to use spike solutions to solve some problems.

According to the XP team, XP was helpful for solving problems due to its iterative nature and spike solutions. When they had a difficult problem to solve, they created a spike solution before implementing the final solution. When they

experienced problems after a release cycle, they solved these in the next iteration of the same release cycle (increment).

Concerning the management of problems, the XP team explain that when a problem occurred in an iteration and it was not resolved, it was passed on to the next iteration where an attempt was made to resolve the problem with a spike solution. If this did not solve the problem, it was passed on as a story card and used in the same iteration. This provided them with enough time and methodological resources to eventually solve their problems.

The XP team state that they had no team related problems.

5.4.4 FDD team

Data warehouse diagram:

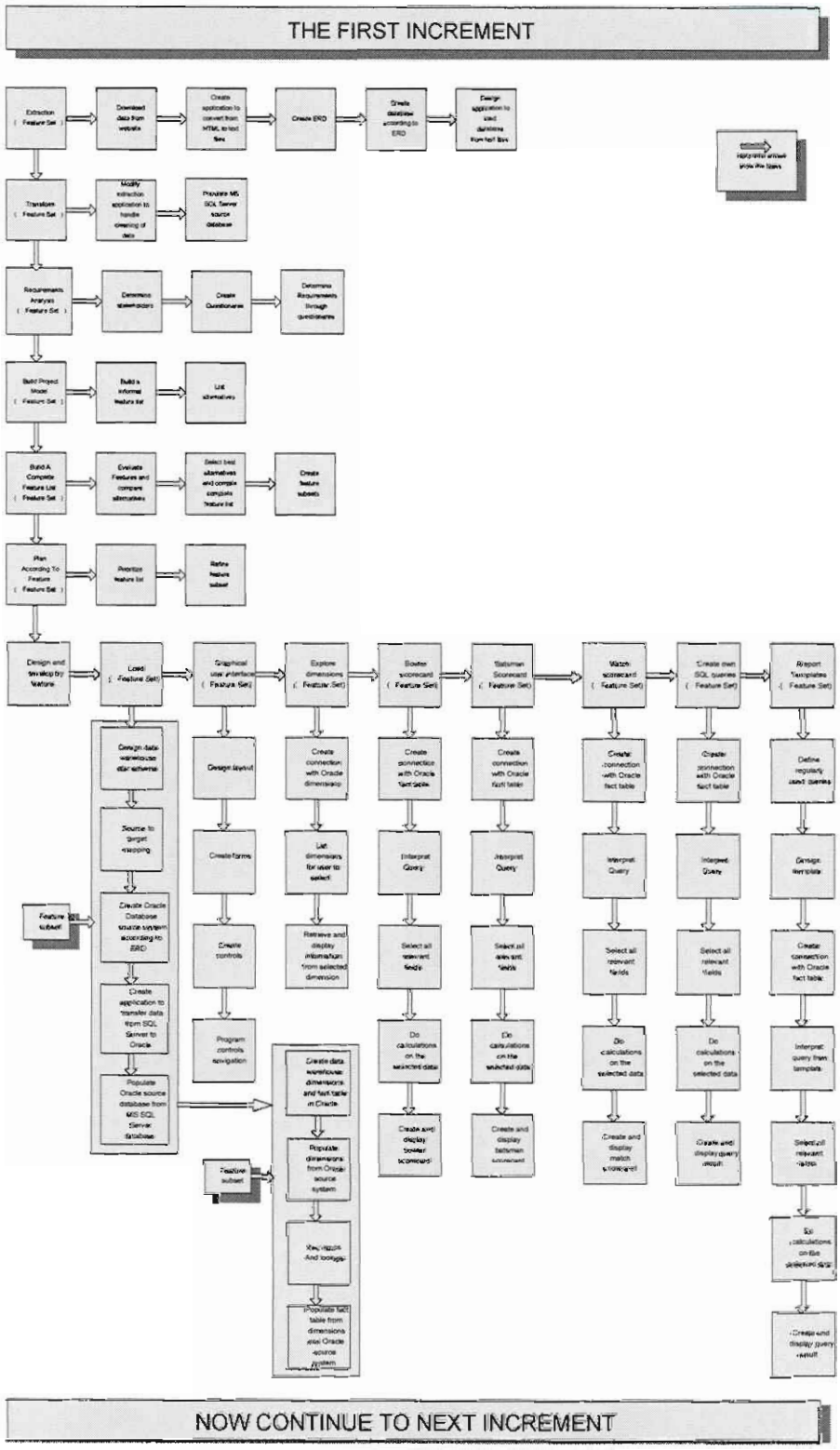


Figure 5.4: Applying FDD to data warehouse development

FDD is uniquely identified by the fact that features are used to capture requirements and to guide the development process, just like the story cards of XP. Features can also be grouped in logical feature sets and feature sub-sets. The arrows in the diagram depicts that a certain feature(s) can only be executed if the pre-defined feature(s) was completed successfully.

***The team's interpretation of FDD's suitability towards data warehouse development before development*:**

The participants had a positive attitude and understanding towards the project by stating: "According to us FDD will work, because it looks almost like the general system development lifecycle, except that development is done in increments and iterations". The answer illustrated that FDD can be applied to data warehouse development because it has the same structure as the normal system development lifecycle of information technology projects.

***The team's interpretation of FDD's suitability towards data warehouse development after development*:**

According to the FDD team, FDD was used effectively in all phases of development where the development of the cricket data mart was viewed as the primary feature set. The data mart was then divided into feature sub-sets (sub-data marts) and further into individual features (or tasks) that needed to be completed in a specific order. The FDD team furthermore stated, "Although FDD worked for developing the data warehouse, it lacked problem solving utilities."

Documentation evaluation:

The FDD team stated that FDD was a very useful methodology for developing a data warehouse because the data warehouse's logical structure was easily

divided into feature sets and feature sub-sets, and identifying individual features was a very easy task. Each data mart can be a feature set and the feature sub-sets (sub-data marts) can be anything from creating the dimensions, creating the fact table, developing the GUI (which in itself can consist of several feature sub-sets), etc. The team found that it was a perfect fit and used it very easily and effectively in the creation of their data warehouse. The team stated that FDD worked very well in all phases of the project. FDD was not used to solve technical and team related problems as the team members were competent enough to solve these problems themselves.

The morning calls (short meetings) supported the FDD team to manage the data warehousing development process in a changing environment. During each meeting, problems were discussed and solutions were determined to solve the problems as fast as possible.

Problems during development

With the development of the overall system, the FDD team encountered a few problems in their project schedule, because the requirements changed as development progressed. Furthermore, only technical issues were experienced.

5.4.5 CC team

Data warehouse diagram:

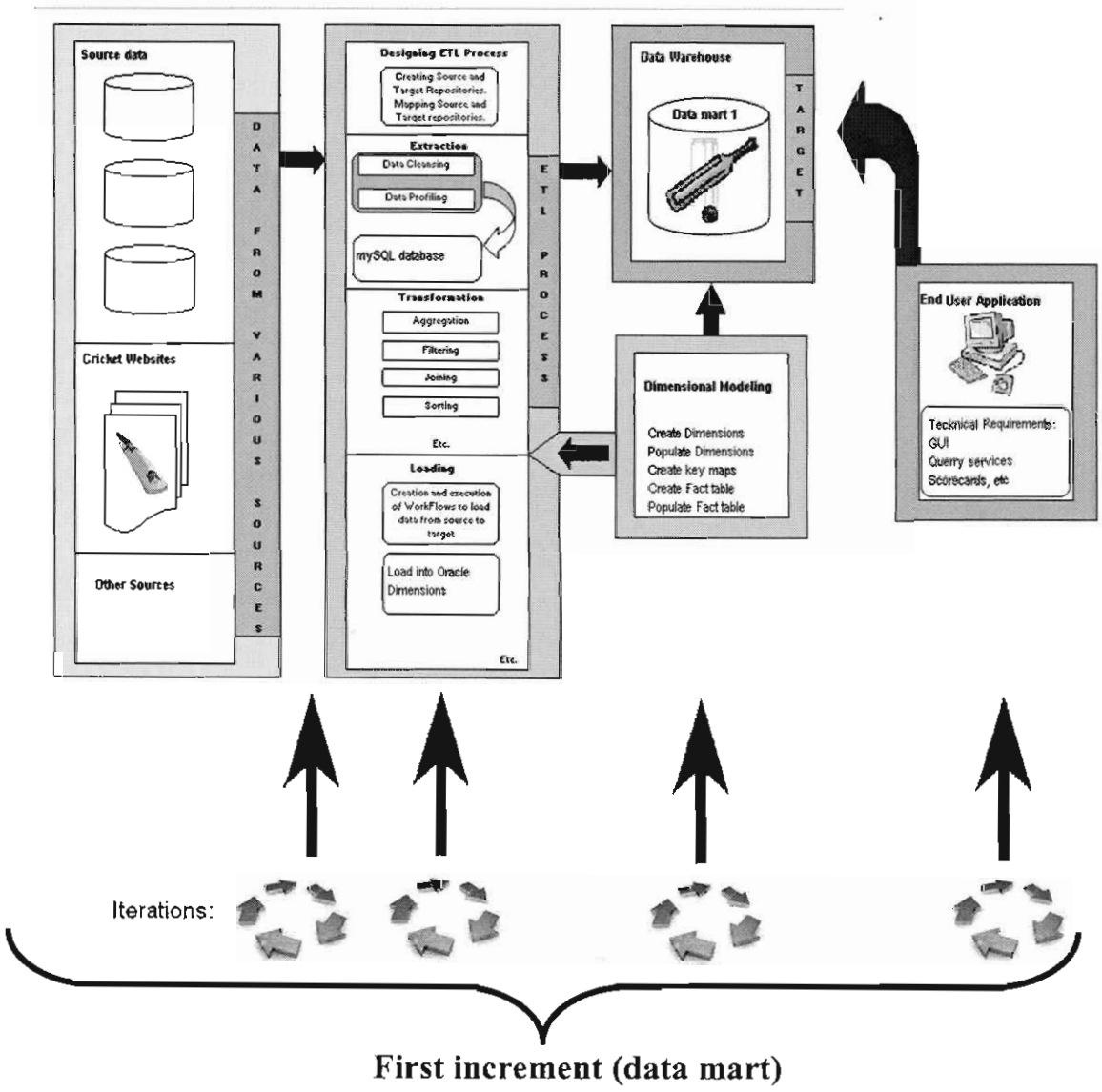


Figure 5.5: Applying CC to data warehouse development

CC is uniquely identified by the seven properties that were applied during data warehouse development. CC is different from other ASDMs, as it does not have a lifecycle that can be followed like that of DSDM or Scrum. Although this team indicated only one data mart as an increment, the teams' projects could practically be viewed as consisting of various increments, since GUI

development can be seen as such an increment.

***The team's interpretation of CC's suitability towards data warehouse development before development*:**

The participants gave a positive response and replied, "Because development is incremental and iterative, it makes it easier to develop data marts as requirements are added."

The answer illustrates that CC has the ability to develop a data warehouse.

***The team's interpretation of CC's suitability towards data warehouse development after development*:**

The CC team explained that CC worked extremely well during the development of their data warehouse. "Because data warehouse development was unpredictable, CC's properties helped create an effective working environment for the team. CC is more a method of communication than a structured methodology."

The answer illustrates that the CC team was more focused on effective communication than the development process.

Documentation evaluation:

According to the CC team, there were no specific steps that had to be followed with CC. However, it provided the team with a few properties that guided them through the project.

- *Frequent delivery:* The team found this property to be very useful in their data warehouse development. Firstly, it helped them break the warehouse down into smaller, more manageable and understandable parts. Secondly, it

helped them keep to a more structured schedule, because they had to make frequent deliveries of the data warehouse and its underlying aspects and functionality.

- *Reflective improvement:* The property enabled them to recover from errors. The team held meetings to discuss what was going wrong and how to correct and later improve it. They also reflected on the work already done and each team member had a chance to express his satisfaction or the contrary. This gave everyone a clear idea of what still needed to improve. The members who struggled with an aspect, were able to ask for help. The CC team found this property almost essential for data warehouse development.
- *Osmotic communication:* The CC team was small, making this property not hard to adhere to, but it will be substantially more difficult with bigger groups. They worked together, and communicated well with one another. Sometimes they kept in contact via e-mail and phone. Every aspect of the data warehouse was communicated.
- *Personal safety:* The CC team stated that it was sometimes important to speak ones mind especially if something was bothering a team member. The personal safety property creates an environment where each team member was encouraged to speak up. They could criticise the work of others and explain if they were unhappy for whatever reason, resulting in a better system or data warehouse being developed. The team members felt safe and at ease working with other team members. However, the team used this property less than all the other CC properties. The team became good friends, being open towards each other. They think that in bigger and unknown participation, this property should get more attention.
- *Focus:* The CC team explained that team members are required to focus on what they are doing. Distractions should be kept to a minimum. This was a very difficult property to uphold, because they had many other priorities. They admitted that this property was unnecessary, as it is somewhat obvious that one needs to focus to make something a success.

- *Easy access to expert users:* According to the CC team the data warehouse was based on some form of activity and process, which, when processed, provides the necessary information. To correctly represent this information, much interaction was needed between the expert users and the development team. The expert users' input will greatly determine the overall presentation of the whole data warehouse. The CC team do however feel that this property was not explored as much as the first five properties. The interaction should be extensive. According to the CC team, the "osmotic communication" property was greatly neglected.
- *Technical environment with automated tests, configuration management, and frequent integration:* The CC team admits that this property was probably least used by them, especially the automated testing which they did manually. They do, however, feel that this property is important, and that it can be used successfully in data warehouse development.

Problems during development

The team states that they had no communication problems, because they used the property "osmotic communication" effectively. Team members contributed peacefully and harmoniously, and, according to them, when they worked, they worked long and hard. Although "automated tests and configuration management" was seen as very important, the team states that this was neglected. This may have resulted in sub-optimal requirement satisfaction. Furthermore, only technical and source data issues were experienced.

5.4.6 ASD team

Data warehouse diagram:

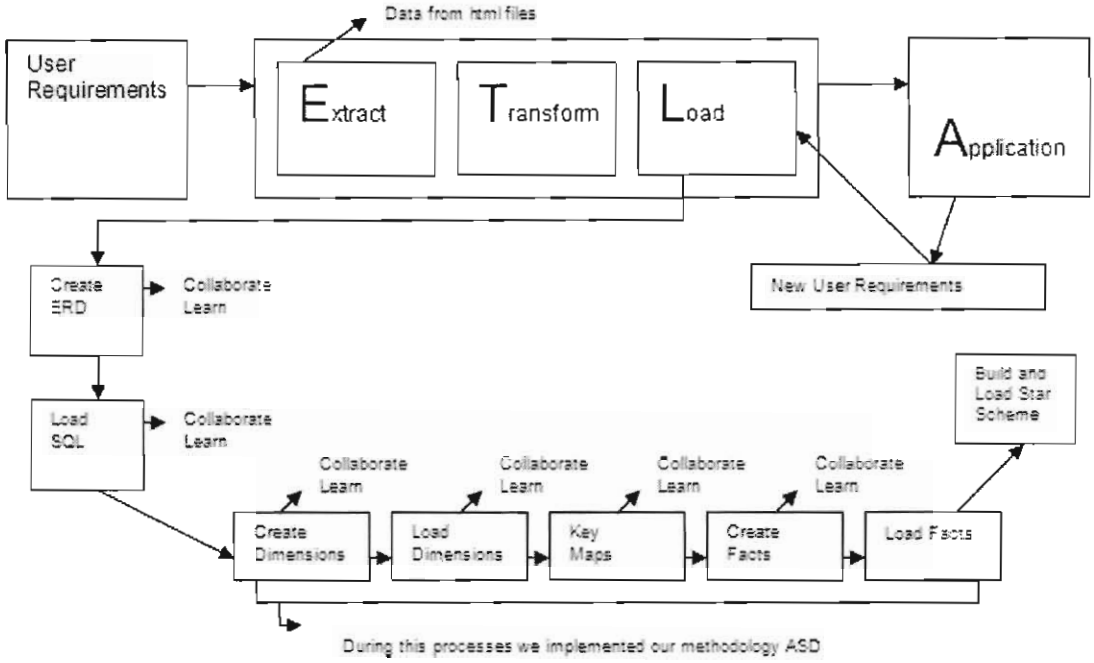


Figure 5.6: Applying ASD to data warehouse development

ASD is uniquely identified by the time boxes and cycles that manage a logical grouping of tasks that has to be developed. Furthermore, ASD sees the development process as a learning process through effective collaboration between team members.

***The team's interpretation of ASD's suitability towards data warehouse development before development*:**

The participants had questions about the suitability of ASDMs on data warehouse development. The team wanted to know if ASDMs in general have been used to develop data warehouses in practice.

***The team's interpretation of ASD's suitability towards data warehouse development after development*:**

The participants emphasise that ASD was successful in the development of their data warehouse. They also explain: "A wonderful aspect of ASD is that every problem is seen as an opportunity and a chance for learning. ASD gave us the space to make our own choices of what needed to be done to make the project a success. ASD guided us in delivering a project on time and within a changing environment."

Documentation evaluation:

Because ASD focuses on adapting to change, the ASD team did their initial project plan, and implemented it in a way that whenever they saw that their initial plan would not work, or when they recognized a problem, they did not hesitate to change it. According to the team, they were very fortunate to receive this ASDM. The ASD team explains that the planning of the technical requirements could easily be changed, where they changed between different Oracle versions, without having major problems. They also argue that ASD allowed them to move cycles around and to adapt to the additional requirements that were not part of the initial project plan.

One disadvantage was that the ASD team feels that they never knew exactly what was expected of them. This resulted in more time spent on planning, rather than implementing their ASDM's prescribed steps.

The ASD team conclude that ASD could be used in the development of a data warehouse because of its adaptability.

Problems during development

The team explains that ASD was very effective because every time they experienced a problem, they were able to change their initial planning. As with

other ASDMs, ASD experienced time schedule problems. The team states that they experienced scope creep problems, where the time available to complete the project was kept the same as the scope expanded because additional requirements were added.

5.4.7 LD team

Data warehouse diagram:

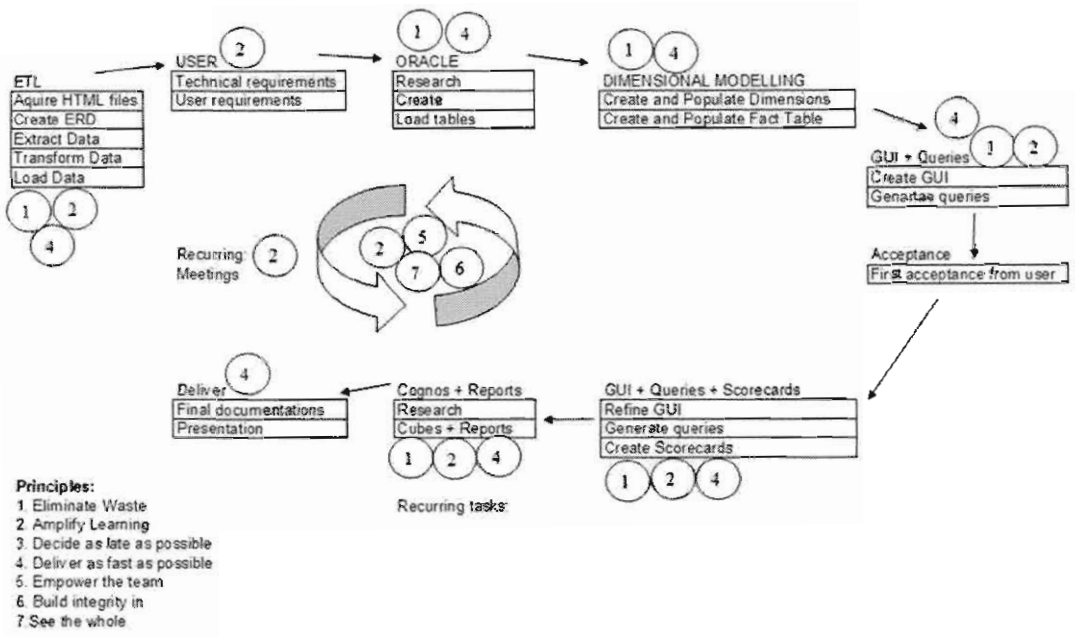


Figure 5.7: Applying LD to data warehouse development

LD is uniquely identified through the seven principles which can be incorporated during the development phases of a data warehousing project. Meetings where held before a logical grouping of tasks were completed to identify problems and to find solutions for the problems during development.

***The team's interpretation of LD's suitability towards data warehouse development before development*:**

The participants were unsure of what was expected from the data warehouse

and replied that they were not sure if LD could be applied for data warehouse development.

***The team's interpretation of LD's suitability towards data warehouse development after development*:**

The attitude and interpretation of the LD team grew positive towards the data warehousing project. They state: "LD worked for us and we think that our warehouse is a success and that LD assisted us in this process. Eliminate waste (principal 1) was our favourite principal, because documentation that was not seen as very important during our data warehouse development project was eliminated. Only important documentation was necessary. A problem we encountered with LD is that it states that users should decide as late as possible what they expect from the system. This did not work for us because we did not have enough time to wait for the user requirements, and had to work with the requirements given to us during the user requirements session."

The answer illustrates that expecting users to "decide as late as possible", is not a good approach when time is restricted.

Documentation evaluation:

The LD team explain the advantages and disadvantages of the seven principles of LD during the development of their data warehouse.

- *Eliminate waste:* Any aspect that did not add value to their data warehouse was eliminated (using the waste tool), including numerous pieces of documentation. The team only had to do documentation of utmost importance, as explained in par. 5.3.2.

  The LD team also experienced delays in approval from users as waste. They explain that they kept developing iteratively to move on within the

confined timeframe of the project without having to wait for approval. They also explain that to save time, no debugging documentation or debug planning sessions were done. If an error occurred, they fixed it. The LD team do, however, state that they sometimes found it difficult to move on to the next iteration when user approval was not received.

- *Decide as late as possible:* The team states that this principle could lead to serious problems because their time to complete the project was limited. They could not wait for users to "decide as late as possible", because the user requirements were given early in the project and the project had a limited time frame.

- *Build integrity in:* The LD team explains that the goal here was to ensure excellent flow of information between customers and the development team. They used "refactoring" as a tool. When a problem occurred, they stopped the development of the specific iteration, and took the time to find and fix the root cause of the problem, before proceeding with further development. However, according to the team this was not always possible, because they sometimes had to continue working to find an alternative solution.

- *See the whole:* According to the LD team, the goal is not to focus on individual parts of the system or data warehouse but rather to focus on the delivering of the data warehouse as a whole. They explain that this can be negative, because they make use of iterations and completing each iteration as fast as possible was a major priority. During the project, the LD team rather focused on the small parts (iterations) to develop the whole data warehouse.

The principles "amplify learning", "deliver as fast as possible", and "empower the team" were applicable to all other ASDMs in the interpretive experiment.

The LD team explain that by continuously working with the customer, no unnecessary features were incorporated in the data warehouse. As a result, they kept the "design simple".

Problems during development

The team explained that after they completed an iteration, they were sometimes uncertain about the correctness of certain aspects of their data warehouse. LD had them moving onto the next iteration without having to wait for user approval. This could have lead to serious problems at the end of the development had there not been enough time to correct major problems.

Furthermore, not all the team members are familiar with the game of cricket on which the data warehouse was based. They used "amplified learning", one of the principles, to help one another understand these concepts by sharing ideas and methods to accomplish certain tasks.

The team explained that when development got confusing, they focused on principle four (deliver as fast as possible). By first getting something going, it was easy to improve on it later.

Motivational problems occurred frequently in the LD team. They had to find ways to motivate and "empower themselves and the team" (principle five). They used the elements mentioned like "feel of progress" to ensure that they worked effectively.

5.5 Data warehouse success

It was clear that all participants in every team had a positive attitude towards the use of the assigned ASDMs to develop a data warehouse. After the evaluation of each ASDM, it was clear that all ASDMs have certain areas, properties or principles that are suitable to develop a data warehouse in a

constantly changing environment. The users created the constantly changing environment by adding development and documentation (only the most important documentation) requirements.

The researcher used specific criteria to verify whether the data warehousing project of each team was successful. The answer to each question is based on the data analysis of the seven interviews, two evaluation sessions and the final project documentation.

The project success criteria consist of the following:

1. A usable project plan with a diagram explaining how the assigned ASDM could be used to develop a data warehouse.
2. Was the data extracted correctly from the cricinfo.com website into the created ERD tables?
3. Was the star-schema (data mart) constructed correctly in Oracle?
4. Was the source-to-target map designed correctly, showing where the star-schema gets its data from?
5. Was the data cleansing acceptable?
6. Was the fact table loaded correctly?
7. Was the connection between the GUI and Oracle data warehouse correct?
8. Was the GUI user friendly?
9. Was it possible to view the fact table and its corresponding dimensions in the GUI?
10. Did the GUI consist of an editable, easy to use query builder?
11. Were the users able to generate scorecards for any ODI or five-day test match (including the first and second innings)?
12. Did the GUI have advanced queries worked out beforehand to help the users interpret the data in the data warehouse?
13. Was Cognos used to build at least one cube (with a drill down operation) and to generate a report?
14. Was the project completed before the deadline?
15. Was the data warehouse effective?

16. Did the team work together effectively?
17. Was the ASDM used correctly?

After analysing all the documentation and other data collected during interviews and evaluation sessions, the researcher determined that all the projects were successful in the interpretive experiment, answering "yes" to all the questions in the above criteria.

The data analyzed in this chapter proves deductions that were made in chapter 4, to be confirmed. In chapter 6 general findings for all ASDMs and unique findings for every individual ASDM will be explained by combining the theoretical deductions made in chapter 4 and the interpretive experiment results (propositions) of chapter 5.

CHAPTER 6

CONFIRMED FINDINGS

6.1 Introduction

In chapter 4 the suitable and unsuitable theoretical characteristics of each ASDM were explained by examining in which phases of data warehouse development a certain ASDM's areas, properties, practices, principles or development phases could be applied to develop a data warehouse in a changing environment. This was done with relation to both Kimball *et al.* (1998) and Inmon's (1996) approaches, resulting in the decision to use Kimball's approach for the interpretive experiment in chapter 5.

In chapter 5 the theoretic deductions made in chapter 4 were practically tested by conducting an interpretive experiment. The teams, participating in the interpretive experiment, were able to develop successful data warehouses using their allocated ASDMs.

After examining the theoretical deductions made in chapter 4 and the interpretative experiment results in chapter 5, these deductions and interpretive results (propositions) will be combined in. This will be done by identifying certain ASDM areas, properties, practices, principles or development phases that could be applied to the data warehouse development phases. Findings regarding the suitability of all ASDMs in data warehouse development will be explained, as well as findings for every individual ASDM.

6.2 Research findings

Firstly, the findings that were applicable to all ASDMs will be explained by combining the theoretical deduction of chapter 4 with the propositions of ASDMs in chapter 5. Secondly, a detailed explanation will follow were every ASDM's theoretical deductions made in chapter 4 will be combined with the

propositions presented in chapter 5.

6.2.1 Findings regarding the suitability of ASDMs in data warehouse development

As confirmed in the interpretive experiment conducted in chapter 5, ASDMs have the ability to be used in data warehouse development. Throughout the interpretive experiment, the users kept the environment agile by adding additional requirements as development progressed. Al ASDM teams stated that communication between team members and stakeholders, as well as effective teamwork were important factors for their data warehouses to be successful.

The source data that was used during the interpretive experiment came from a real world source and data inconsistency problems could not be associated with the academic setting of the interpretive experiment where ASDMs were used to develop data warehouses. With this said, all teams, except XP, explained that ASDMs lack capabilities to solve technical issues during data warehouse development.

All ASDM teams further explained that the project plan and schedule helped them to get their minds organised regarding what had to be done next. All teams also stated that additional requirements could be implemented into the evolving data warehouse without any major problems, because of graceful changeability of star-schemas (Kimball et al., 1998:148). All these deductions were confirmed in chapter 5, including applicable propositions of all ASDMs.

6.2.1.1 Collecting requirements

It was possible to use ASDMs in the collecting requirements phase during the interpretive experiment, because Kimball *et al.* (1998) and ASDMs collect requirements before development commences. Furthermore, ASDMs were

used because they follow the same approach as Kimball *et al.* (1998), where satisfaction of user requirements is of the utmost importance.

ASDMs in the data warehousing projects ensured frequent incremental delivery that resulted in early user identification, continuous user involvement, earlier requirements identification, and feedback on the implemented requirements. ASDMs explained that any means possible could be used to collect requirements – that is the reason why interviews (as explained by Kimball *et al.*, 1998:97) were used to collect significant information to develop a data warehouse of value in chapter 5. ASDMs ensured that users where always part of the project (i.e. partially available during interviews and evaluation sessions) to keep developers on track to satisfy user requirements.

ASDMs seemed to have a problem with the time schedule where requirements were changed or new requirements added. ASDMs would be more applicable to any project if a technique was used to manage the time schedule as well as the work-breakdown structure when requirements are changed or added in a project.

6.2.1.2 Data modelling

ASDMs do not specify which data model should be used, but they do explain that projects have different characteristics that should be accounted for, and during a data warehousing project a star-schema was required to model the requirements into diagrams. For this reason the teams in the interpretive experiment used star-schemas that were developed from the ERD tables using source-to-target maps. Another reason a star-schema was used, is that it has the trait of graceful changeability (Kimball *et al.*, 1998:148) allowing changes to be made easily to the star-schema as requirements are added.

The deduction regarding unsuitability made in par 4.2.1.2, where the researcher explained that ASDM projects have not been proven to be successful when

using star-schemas, is false, as it was confirmed to work in the interpretive experiment for every team.

6.2.1.3 Data staging

The confirmed theoretical deductions and propositions of ASDMs during data staging will be explained in the following paragraphs.

The ETL-process of every team's data warehouse was represented by the development process of every ASDM. Development was done in increments (including the ETL-process) where every increment was seen as a data mart or a sub-data mart; including the diagram of the requirements into a star-schema; the design and data staging of the star-schema; the GUI associated with the data mart; and the reports that were generated from the data mart. Each increment or sub-data mart was developed iteratively. The interpretive experiment confirmed that using ASDMs in data warehouse development, a data mart (for CC) or sub-data mart (for all other ASDMs except CC) was developed in iterations, sprints, cycles, and feature sub-sets, depending on the ASDM. The data mart's logical structure (including everything from collecting requirements to GUI and report generation) was easily divided into individual tasks, iterations, sprints, cycles or features. Because development was done iteratively, problems were identified easier and solved quicker, which resulted in an improved data warehouse design as iterative and incremental development progressed. New and additional requirements were added without any major problems.

The teams could also use tools and techniques from other methodologies to get the job done. The technique used by almost all teams during data warehouse development was pair programming. The ASDMs kept the users actively involved during the development process, and the teams were empowered to make their own decisions to satisfy the user requirements that were given during the requirements session. Development was reversible where ASDMs,

with the exception of FDD and CC, were applied to data warehouse development as participants could backtrack to the last safe point in order to start a new development approach.

Every increment or sub-data mart was delivered incrementally to create the data mart, whereafter the data mart (including everything from collecting requirements to GUI development and report generation) was deployed as a whole to the users. ASDMs encouraged and improved development of a data warehouse in an agile environment.

6.2.1.4 Data access and deployment

Before the data mart of every team was deployed, it was thoroughly tested to identify whether the primary requirements as specified in the requirements session were fulfilled. Testing was also integrated throughout the development process for all ASDMs, except in the case of the FDD and CC teams. After successful testing results, the data mart was implemented and the use of the data warehouse was transferred to the users. Every team trained the users to use the data warehouse effectively by using the evaluation sessions as well as the mini-manual the participants created for the final evaluation session.

The successful completion of every team's data warehouse confirms that frequent delivery and constant integration is essential to project success. The interpretive experiment confirmed that a data warehouse could be delivered in increments to create the data mart, where the data mart was deployed as a whole to the users.

6.2.2 Findings regarding the suitability of DSDM in data warehouse development

Iterative sub-data mart development worked throughout the development phases of DSDM where the data warehouse was developed. The team that

used DSDM for their data warehouse explained that DSDM would only work if functionality requirements were set. The reason for saying this was that the environment was kept agile by adding additional requirements. This caused the team to miscalculate the duration of their individual tasks that had an impact on the functionality, because the deadline for the project was fixed. The addition of extra requirements may also have resulted in a project that was not completed on time, as participants in the DSDM team explained that the functionality in their project decreased because of the extra requirements that caused individual time boxes to change.

A suitable technique could also be included in the DSDM methodology as the team struggled to divide the workload among members.

6.2.2.1 Collecting requirements

The following deductions and propositions were confirmed during the interpretive experiment.

DSDM recognises that facilitated sessions can be used, as explained by Kimball *et al.* (1998:97). The requirements session held by the users could be viewed as a facilitated session. DSDM could thus be used effectively in the collecting requirements phase of Kimball's approach.

During the data warehousing project, the business study resulted in early user involvement as well as early requirements identification. Because DSDM does not focus on documentation, prototypes were used to capture information and requirements. DSDM was suitable to gather data warehouse requirements, since it is a people-oriented methodology, where user requirement satisfaction was the primary focus, as is the case with Kimball's approach.

6.2.2.2 Data modelling

DSDM includes a logical model that was used by the participants to implement a star-schema (data mart). The business area was defined by the affected business processes, and because a star-schema represents a business process, the process (ball for ball data in cricket tests) and requirements could be modelled into a star-schema. The star-schema, source-to-target map, and source ERD tables were used without any problems in the data warehousing project. The deduction made in par 4.2.2.2 that DSDM only uses six core techniques was confirmed to be false. Techniques such as star-schemas, source-to-target maps, and ERD tables were used to develop a data warehouse with DSDM.

These were the findings and confirmed theoretical deductions presented by DSDM in the data warehousing project.

The following findings and theoretical deductions were confirmed by DSDM.

6.2.2.3 Data staging

DSDM is suitable for successful data staging. The ETL-process was done iteratively where requirements could be added and adjusted as development progressed. Although the added requirements had an impact on the functionality and individual task estimates, the iterative ETL-process ensured that flows were detected easily and that requirements could be added or modified.

The function model iteration phase of DSDM was combined with the first step of Kimball's data staging process by using a source-to-target map as the schematic format that contains indications as to where the data has originated.

Not all the tools were used during the data warehousing project as explained by Avison and Fitzgerald (2003:286). The only tools that were used effectively

during this project were Oracle, Visual Basic, and C#.

6.2.2.4 Data access and deployment

DSDM methodology explains that the output of the implementation phase is a user-manual that can be used to train the users to use the data warehouse effectively. Because documentation was kept to a minimum, the participants completed a mini-manual and presented their data warehouse during the two evaluation sessions to train the users in using the data warehouse effectively. Incremental development and deployment was used effectively to complete, combine, and test the developed prototypes before delivering the data warehouse to the users. The data warehouse was tested throughout the development process.

The output of the "design and build" phase of DSDM was fulfilled, because a tested data warehouse that met at least the most important requirements was delivered to the users.

No incremental review documents or project review documents were developed by the participants as required by DSDM, because the data warehouse was kept "lite" by removing some of the less important documentation.

During the evaluation of the ASDMs toward data warehouse development, DSDM presented the most unsuitable characteristics to be applied in data warehouse development. The main reason for stating this is based on the fact that when requirements are added, it compromises the duration of individual tasks that may result in a project that is not completed in time, or a project with diminished functionality.

6.2.3 Findings regarding the suitability of Scrum in data warehouse development

The unique characteristic of Scrum where development was done in sprints was viewed as a very important component of the Scrum team's data warehouse success. The team explained that because the workload could be divided into sprints, the project was completed much easer. Every sprint was viewed as a logical grouping of tasks that was completed before the next logical grouping of tasks was started. This resulted in newly identified technical and user requirements that made the data warehouse even more effective. Furthermore, the daily meetings improved communication where team members knew exactly what to do during the current and future sprints. These were deductions confirmed by the interpretive experiments.

The Scrum team did, however, explain that they had a loss of initiative and great ideas because of the changing environment. In the interpretive experiment it was confirmed that a team should not spend too much time on initial planning because when the environment changes, it causes redundant data containing an unnecessary number of inconsistent dates and tasks.

Using Scrum to develop a data warehouse, the following deductions and propositions were confirmed by the interpretive experiment (see par. 6.2.3.1 – par. 6.2.3.4).

6.2.3.1 Collecting requirements

The fact that Scrum is a requirements driven methodology made it suitable for application during the collecting requirements phase of Kimball's data warehouse, as Kimball encourages requirements collection before development takes place. The primary requirements that were given during the requirements session were listed and prioritized in the product backlog.

During each pre-sprint planning phase the identified requirements were extracted from the prioritized backlog and moved to the sprint backlog to be completed during the next sprint. In this way the requirements that were identified in the requirements session and interviews were satisfied in sprints of approximately twenty days in duration. Meetings of approximately 5 minutes were held every day to ensure that requirements were met as development progressed. After every sprint a post-sprint meeting was held to determine whether all the selected requirements were fulfilled, and to identify solutions for possible problems that may have occurred during the sprint. New and updated technical and user requirements gained from the post-sprint meetings were added, updated, and prioritized in the product backlog as development progressed.

The statement made in par. 4.2.3.1 where Scrum was explained by focusing more on team empowerment than collecting requirements, is confirmed to be false, as satisfying user requirements are of the utmost importance to the Scrum methodology. Although Scrum focuses on the management of the development process, it also focuses on the satisfaction of user requirements.

6.2.3.2 Data modelling

Although Scrum did not mention a star-schema as a data modelling technique (as explain in par 4.2.3.2), a star-schema was confirmed to work in a data warehousing project where Scrum was applied.

The deduction made in par. 4.2.3.2 explaining that key pieces and technological requirements can be estimated during the first sprint, was found to be false, as the first sprint was used to extract the adequate data for the data warehousing project. The technological requirements were known before development started as a data warehouse had to be developed using Oracle, Cognos and programming languages such as VB and C# to clean the data and build a GUI. The installation of Oracle was not included in the project as a

sprint.

Because requirements have the tendency to change, the updated requirements were modelled into the star-schema using the graceful changeability of Kimball *et al.* (1998:148) and iterative development of Scrum.

6.2.3.3 Data staging

Scrum recognises that every project has its own characteristics and degree of uncertainty that made the manually designed data staging tools applicable during the ETL-process of a data warehousing project.

The ETL-process was done over three sprints (see diagram 5.2). During sprint 2 (as illustrated in diagram 5.2) the ERD tables were loaded from the text files that had to be cleansed. During sprint 3 the dimensions of the star-schema were created and populated. Sprint 4 was the last sprint of the ETL-process where the fact table was loaded using the created source-to-target maps. After the successful completion of these sprints, the star-schema was loaded.

As deducted from literature, the experiment confirmed that (in par. 4.2.3.3) the ETL-process can be executed using a product backlog and sprints. To keep track with the ETL-process, the Scrum team had daily 5 minute meetings to identify and solve problems. Post-sprint meetings were held after each of the three sprints to determine whether the iterative ETL-process was successful. If the sprint or an iteration in the sprint was not successful, it was repeated. If the sprint was successful, the next sprint commenced by extracting the required requirements from the product backlog.

The Scrum team assigned a different sprint leader to every sprint. This technique allowed a diverse development approach that resulted in a project with diverse characteristics. As the team members grew to know one another, the duration of the daily and post-sprint meetings declined, as problems were

solved quicker by understanding each team member's point of view faster. The sprint allowed the team members to deliver a sub-data mart in twenty days or less, and not 30 days as specified by Scrum.

6.2.3.4 Data access and deployment

Because a sprint was done iteratively, it was possible to add or modify requirements by using a simple iteration. The Scrum team used the built-in implementation phase to deliver the data warehouse. During this phase the users were trained to use the data warehouse effectively by submitting a mini-manual and using the evaluation sessions to present their data warehouse – thereby supporting the deduction made in par. 4.2.3.4. This holds that a training program can be made during the planning sub-phase of the pre-game phase during which team members plan and prepare their presentation and mini-manual for the final evaluation session.

During the evaluation of the Scrum team's data warehouse the users were very satisfied as the diverse team presented the best project of all the ASDMs. The team was very diverse as some team members were detail oriented by keeping the team on schedule, while the others used pair programming to design their data warehouse.

6.2.4 Findings regarding the suitability of XP in data warehouse development

The XP methodology was used effectively by the XP team to deliver a very satisfactory data warehouse. The results of the interpretive experiment enriched the understanding of the use of XP in data warehouse development.

Problems in iterative development were resolved early on using XP, since they were passed on to the next iteration where an attempt was made to resolve the problem with a spike solution. If this did not solve the problem, it was passed on

as a story card and was added in the current iteration. This technique provided the team with enough time to solve all their problems as development progressed.

XP was the only ASDM that supported the team when they experienced technical problems by having the option to use a spike solution. Because development was done iteratively, they could solve a difficult problem by creating a spike solution before implementing the final solution. When problems were experienced during a "release cycle", they were able to solve it in the next iteration of the same release cycle.

The interpretive experiment confirmed the theoretical deductions made in chapter 4.

6.2.4.1 Collecting requirements

XP has a unique characteristic of collecting requirements using story cards. Because XP focuses on communication between the team members and the users, interviews and a requirements session were used to collect requirements. The requirements collected during the requirements session and interviews were written on the story cards by the team members. It was not necessary for the users to complete the story cards, as the requirements were thoroughly explained during the requirements session and interviews. Thus, the team members completed the user story cards where the users explained exactly what they wanted.

This proved the deduction made in par. 4.2.4.1 to be false that users have to write user requirements on story cards and that requirements can only be collected using story cards and not interviews. Thus, story cards can be used in conjunction with interviews and a requirements session during an XP data warehousing project. The team members also completed their own story cards that contained technical requirements, as it were their own responsibility to

incorporated technical requirements.

The story cards guided the development as the team members could organise their minds regarding the requirements which had to be satisfied next. Because XP focuses on effective communication between team members and users, the story cards simplified listing as team members knew exactly what was expected from the data warehouse.

The deduction made in par 4.2.4.1 that users will always be on-site, is therefore impractical as users are typically managers and only had time available during interviews and evaluation sessions.

6.2.4.2 Data modelling

The XP team kept their design simple by diagramming the collected requirements into a star-schema. "Refactoring" was accomplished through using star-schemas that improved communication between team members where each team member knew what was expected. Furthermore, the star-schema was used to organise and model the requirements into an organised fashion where graceful changeability of Kimball *et al.* (1998:148) was applied. These were deductions confirmed during the interpretive experiment.

6.2.4.3 Data staging

The unique characteristic of XP is "pair programming" where the team members identified errors faster during the transformation and data staging process, creating a learning environment for the team members. "Designing" (one of the four activities of XP) was used during this data warehousing project to incorporate new and updated designs into the existing data mart. The "metaphor" (another of the four activities of XP) guided the design and fulfilled user requirements where team members transformed and loaded data that was of value to users. "Testing" was incorporated throughout the ETL-process to

ensure that the process was followed correctly and that the data staging program code does what it was supposed to do. Furthermore, the quality of the data staging program code was increased by using "collective ownership", and duplicated data of no value was removed by applying the core practice, "refactoring". These were all deductions that were confirmed in chapter 5.

The assumption made in par 4.2.4.3 that XP projects are only partially adapted in most organisations seem be true, as not all practices of XP seem to be applicable towards data warehouse development, as confirmed in par 5.4.3. "Test driven development", "customer test" and "on-site customers" could only be performed partially because of the lack of consistent everyday user involvement. XP is built on user involvement, making "testing" and doing "quality assurance tests" user dependent. The "planning game", one of the practices of XP that was partially used as the academic setting of the project, made it difficult to practice the planning game.

6.2.4.4 Data access and deployment

The XP team used "small releases" to implement the sub-data mart they developed into the existing data mart. Before every sub-data mart was implemented, it was thoroughly "tested" (one of the four activities of XP), becoming part of the data mart as a whole that was deployed and presented in the final evaluation session. The data warehouse was kept "simple" (one of the four values of XP) by the team by training the users, using a submitted mini-manual and presentation during the final evaluation session. These deductions were confirmed in the interpretive experiment.

The XP team does, however, state – and it is confirmed in other projects, that XP can be used in correlation with other methodologies and ASDMs. The technique "pair programming", which is identified when using XP, was used by several other ASDM teams. Except from the problems they experienced from the academic setting of the project, the XP team explained that they would

definitely use XP in future data warehousing projects.

6.2.5 Findings regarding the suitability of FDD in data warehouse development

FDD has the unique characteristic of developing in features, feature sub-sets and feature sets. In this project a feature set was equivalent to a sub-division of a data mart, called a sub-data mart, while a sub-feature set was equivalent to a sub-division of a sub-data mart (see diagram 5.4). Features were viewed as requirements that were gained from the requirements session and interviews that had to be completed in a specific order. FDD was applicable to data warehouse development, but the FFD team explained that FDD lacked problem-solving abilities.

The FDD team encountered problems in their project schedule as development progressed, because requirements changed and new requirements were added.

The following paragraphs summarize the confirmed findings of using FDD in data warehouse development.

6.2.5.1 Collecting requirements

Because FDD did not explicitly state which data collection techniques should be used, interviews and a requirements session were chosen and used successfully during the collecting requirements phase of Kimball's approach. These features were then prioritized in a feature list and grouped in feature sets and feature sub-sets (as seen in diagram 5.4) where each logical feature grouping represented a specific domain (i.e. sub-data mart) within the data mart. There were no problems during the requirements collection phase of Kimball's approach.

6.2.5.2 Data modelling

The "domain object modelling" (best practice of FDD) was used by the FDD team to develop a star-schema where requirements could be added or adjusted by using graceful changeability of Kimball's approach. During the "design by feature and build by feature" phase, the requirements were represented and further planned in detail by using the star-schema. The star-schema was confirmed to be effective in the interpretive experiment by using FDD to develop a data warehouse.

6.2.5.3 Data staging

FDD was found to be applicable to the ETL-process and data warehousing project as a whole, because the data warehouse's logical structure was easily divided into feature sets and feature sub-sets, including anything from creating and loading dimension tables to the creation of the GUI. The team found this to be an affective technique for work division and to get the team to understand what was expected from them.

The deduction made in par. 4.2.5.3 that the ETL-process can be done during the "design by feature and build by feature" phase where the features can be planned, built, tested, and iteratively integrated into the existing data mart, was confirmed in the interpretive experiment. As part of the "design by feature and build by feature" phase, a feature set (increment) was selected, designed, and tested to become part of the existing data mart. Furthermore, the deduction that the ETL-process was logical and has no process pride was also confirmed by the interpretive experiment.

The morning calls (short meetings held every morning) helped the FDD team to manage the data warehousing development process. During each meeting problems were discussed and solutions determined to solve the problems as quickly as possible. The meetings also helped the team to change and add requirements to the data warehousing project without any major problems.

Although FDD does not mention any tools that could be used to clean data, manual data staging tools were written that worked effectively in the FDD team's data warehouse.

6.2.5.4 Data access and deployment

The deduction made in par. 4.2.5.4, stating that features can be iteratively designed, tested, and integrated to deliver a sub-data mart incrementally to form the data mart (data warehouse) and then deploy the data mart as a whole, was confirmed by the interpretive experiment. Furthermore, the "reporting" practice of FDD supported the team members in delivering a data warehouse of value to the users. The users were trained in using the data warehouse during the evaluation sessions and by submitting a mini-manual.

Although the team struggled to stay on schedule as the environment changed, FDD was used successfully in developing the data warehouse. The changing schedule can be managed by implementing an additional time frame after every increment (feature set), which can be used in cases where certain requirements are changed or where new requirements are added during feature set (incremental) development.

6.2.6 Findings regarding the suitability of CC in data warehouse development

The CC team was the only data warehousing team that made an increment equivalent to one data mart where every data mart was developed iteratively. Different aspects of the data mart were delivered iteratively during the development process to form a data mart. The data mart was deployed as a single increment to the users.

CC worked extremely well during the data warehousing project, as the CC

properties supported the team in creating an environment where members could work together effectively. It was confirmed in the interpretive experiment that CC is more a method of communication than a structured methodology in which certain development phases had to be executed to complete a project successfully.

The property "osmotic communication" was not hard to adhere to as the team was small and the workspace small enough for the team to constantly stay in contact. The "personal safety" property created an environment where each team member could state his/her point of view without feeling threatened. This resulted in a more effective working environment where requirements could be fulfilled easier and faster. The team members did not experience any team related problems because "osmotic communication" was present throughout the development process.

The CC team did, however, feel that it was unnecessary to view the property "focus" as a property, as it is common sense that all team members must be focused on the goal to make any project a success.

6.2.6.1 Collecting requirements

The deduction made in par. 4.2.6.1 that "osmotic communication" encourages developers to fulfil the requirements, and to ensure that team members understand what is expected from the project, was confirmed during the execution of the interpretive experiment. Because CC focuses on communication and the satisfaction of user requirements, interviews and requirements session were used to effectively collect requirements during the first phase of Kimball's approach.

6.2.6.2 Data modelling

Although CC did not define any techniques for modelling requirements into

diagrams, the use of a star-schema was found to be very effective in data warehouse development where CC was used. CC understands that every project has its own characteristics, and explains that team members are able to use their own tools and techniques to get the job done. This is the reason why star-schemas were used, as it is a unique characteristic of the data modelling phase of Kimball's approach.

6.2.6.3 Data staging

The deduction made in par 4.2.6.3 was confirmed by the interpretive experiment where the CC team used "frequent delivery" and incremental (data mart) delivery during the ETL-process. "Frequent delivery" supported the CC team by breaking the data warehouse development down into smaller, more manageable and understandable parts. "Frequent delivery" also supported the team in keeping to a more structured schedule, because the sub-data mart was completed iteratively.

"Reflective improvement", one of the properties of CC, enabled the team to recover from errors that were experienced during the data staging process. Meetings were held to discuss what went wrong and how to solve these identified problems. They also reflected on the work already completed, and each member was granted a chance to enhance the project design.

Although CC does not state whether tools or newly developed programs can be used to clean data, it was confirmed that manually written data staging tools were used effectively by the CC team. These were the propositions and deductions that were confirmed in chapter 5.

6.2.6.4 Data access and deployment

The team found the property "frequent delivery" very useful as increments of the data mart were delivered frequently to create the data mart that was

deployed as a whole to the users during the final evaluation session.

The deduction made in par. 4.2.6.4 where the technical environment was used to test and control tasks by merging changes during automated tested (done manually as explained in par. 5.4.5 under documentation evaluation), configuration management (less used by CC team) and frequent deliveries, was confirmed by the interpretive experiment. Testing was a constant process throughout the development process.

Although CC does not explain a specific method of implementing a data warehouse, it was found to be effective by implementing a data mart as a whole.

CC was used effectively as it related to the data warehouse lifecycle of Kimball *et al.* (1998) without any major problems. The seven properties of CC were incorporated throughout the data warehouse development phases (some more than others), making CC applicable in data warehouse development. The deductions made in par 4.2.6.1 that CC may be too small for a large data warehousing project was found to be false, as the CC team delivered a data warehouse of satisfactory usability.

6.2.7 Findings regarding the suitability of ASD in data warehouse development

The unique characteristic of ASD is that every problem is seen as a learning opportunity. The ASD team explained that ASD gave them the freedom to make their own choices toward ensuring the project's success. ASD guided them to deliver a data warehouse on time within a changing environment.

ASD was very effective – whenever the team experienced a problem, they were able to change their initial planning to adapt to the changing environment the identified problem caused. By changing the initial planning process and by

finding solutions for the experienced problems, a collaborate working environment was created.

A proposition deducted from the interpretive experiment was that the ASD team spent too much time on the planning process, because according to them they were not exactly sure what was expected after the requirements session took place.

The following paragraphs summarize the confirmed findings of using ASD in data warehouse development.

6.2.7.1 Collecting requirements

Although ASD explains JAD sessions to collected requirements, a requirements session (conducted in JAD format) and interviews were used to collect requirements effectively before and while development took place. During the speculation phase, the team recognized that it would be acceptable to collect requirements using interviews and a requirements session instead of JAD sessions.

Although ASD did not support the solutions of technical problems, the ASD allowed the team to change their technical requirements easily where they had to shift between different Oracle versions. Although ASD was adaptive during development, the team experienced scope creep problems when new, time consuming requirements were added within the existing time frame.

During the "project initiation" step the data warehouse requirements were identified, after which the project time boxes were determined based on the gathered requirements. The team worked together effectively as collaboration was good and the team knew how to satisfy the gathered requirements. Furthermore, the "learning loop" was used to identify new technical and user requirements as development progressed. These findings supported the

theoretical deductions made in chapter 4.

6.2.7.2 Data modelling

During the "speculation phase" (step 1 of adaptive cycle speculation), the team determined that a star-schema had to be used to model the collected requirements into an understandable format for their data warehousing project. The star-schema was confirmed to be effective when using ASD as an ASDM, although ASD focuses on collaborate teamwork and regards every problem as a learning activity to create opportunity.

6.2.7.3 Data staging

The ETL-process was executed by ASD during data warehouse development without any major problems. The development process of ASD is uniquely identified by the time boxes that guide development. The time boxes supported the team in dividing the logical structure of the data mart into manageable sub-data marts. Each sub-data mart was assigned a fixed delivery time during which a logical grouping of requirements were satisfied.

The deduction made in par. 4.2.7.3 was confirmed during the interpretive experiment, i.e. that the team learned the most by transforming the data to determine its capabilities and whether the users would be satisfied with the data provided.

During the adaptive lifecycle activities, "project initiation" and "adaptive lifecycle planning" were represented by planning what data should be extracted by the team to fulfil the identified requirements. The transformation process was represented by "component engineering", and the "quality and final Q/A and release" was preformed iteratively where requirements were updated as the ETL-process progressed. These deductions were confirmed in the interpretive experiment where ASD was used to develop a data warehouse.

6.2.7.4 Data access and deployment

Although ASD focuses on collaboration, speculation, and learning activities, the data warehouse was delivered incrementally using cycles to create the data mart, whereas the data mart was deployed as a whole to the users during the final evaluation session.

The status of the project was always known as the data mart was tested throughout the development process. "Final Q/A and release" (explained in par. 6.7.3) was also used during the deployment phase of the data warehousing project. Maintenance and testing were used during the learning phase as the team learned by testing and monitoring the data warehouse for deficiencies and by reviewing the technical quality of the data warehouse. The "learning loop" was created in this manner by monitoring and testing the data warehouse. These were deductions of chapter 4 that were confirmed by the interpretive experiment in chapter 5.

Although the deduction made in par. 4.2.7.4, that ASD is primarily focused on dealing with collaboration and concurrency, was confirmed, it is also proposed that ASD is effective in the deployment of a data warehouse.

ASD was applicable to data warehouse development as a very satisfactory data mart was deployed to users. Although scope creep was experienced as the environment grew more agile, ASD was very adaptive when requirements were added.

6.2.8 Findings regarding the suitability of LD in data warehouse development

LD does not have a specific lifecycle that is used to develop a project successfully. LD only explains seven principles and 22 tools that can be used

by any project in an agile environment.

LD was used effectively during the data warehousing project. The team viewed the principal "eliminate waste" as the principal with the most value, causing documentation that was seen as not very important to be eliminated. The interpretive experiment confirmed that to "decide as late as possible" (principle of LD) could result in project failure, as data staging takes up almost 80% of the time. Data staging is most successful if requirements are collected before the process begins. If team members had to wait for requirements, the development process of the data warehouse was delayed.

The team used the principle "amplified learning" because all team members were not familiar with the game of cricket. This principle allowed them to put some time aside to learn the game, as the whole data warehouse was based on cricket.

The team also explained that because the project was so large and time consuming, they had to motivate and "empower themselves".

The principles "amplified learning", "deliver as fast as possible", and "empower the team" were applicable to all other ASDMs in the interpretive experiment.

6.2.8.1 Collecting requirements

LD explains that team members could use their own techniques to gather requirements. Thus, interviews and a requirements session were adequate to gather the primary and technical requirements needed for developing the data warehouse.

The principles "delay commitment" (including "lean thinking") and "decide as late as possible" were confirmed to have a negative impact on the interpretive experiment as development was delayed because team members waited for

requirements from users. Team members could not wait for users to make up their minds, as the project timeframe was limited. These deductions were confirmed in the interpretive experiment.

6.2.8.2 Data modelling

LD does not explain any form of star-schema or ERD tables, but explains that every project has its own characteristics, and emphasises the fact that team members can use their own tools and techniques to complete the project. Although LD does not define any techniques for modelling requirements into a star-schema, the use of star-schemas were found to be very effective in data warehouse development where LD was used. The deduction made in par. 4.2.8.2 that star-schemas will emerge if Kimball's approach is used, was confirmed in the interpretive experiment

6.2.8.3 Data staging

Deductions confirmed in chapter 5 will be explained in the following paragraphs.

The deduction made in par. 4.2.8.3 that LD's principles can be integrated throughout the project, was found to be false as "decide as late as possible" and "delay commitment" were not proven to be applicable in the data warehousing project.

"Eliminate waste" was used during the ETL-process where duplication fields and unnecessary data were deleted. The LD team also viewed delays in user approvals as waste. No debugging documentation or debugging planning documentation was compiled in order to save time during the data warehousing project.

The principle "build integrity in" was used during the ETL-process, i.e. the team

used refactoring tools as well as manually written data staging tools to transform the data. When problems occurred during the development process, they were fixed immediately without documenting changes and new approaches.

The ETL-process was conducted iteratively to deliver the increment (sub-data mart) as fast as possible. This resulted in a data mart that was deployed as fast as possible to the users.

6.2.8.4 Data access and deployment

LD does not explicitly explain a way of how a data warehouse should be deployed. It does, however, specify that deliveries should be done quickly. The LD team delivered sub-data marts to create a data mart as a whole, which was deployed to the users during the final evaluation session. Quick delivery (deduction made in par 4.2.8.4) was confirmed during the interpretive experiment, i.e. value was added as fast as possible to the LD data warehouse, not allowing delays during testing, integration and deployment . The LD team explained that when they got confused, they focused on "delivering as fast as possible", because having at least started, actions could always be improved upon at a later stage.

The users were trained to use the data warehouse effectively during the two evaluation sessions and the submitted mini-manual. Furthermore, the LD team achieved "perceived integrity" (deduction made in par 4.2.8.4) by testing and implementing the correct requirements after the first evaluation session.

The team experienced the principle "see the whole" as negative, because development was done in increments and iterations, and each increment had to be developed as fast as possible. The LD team explained that they focused on the completion of one increment (sub-data mart) at a time to complete the whole data warehouse successfully.

LD (like XP) was only partially adapted in the data warehousing project. Some principles proved to be applicable to data warehouse development, such as "deliver as fast as possible", "eliminate waste", and "build integrity in". Other properties such as "decide as late as possible", "delay commitment", and "see the whole" were found not to be applicable to the data warehousing project.

6.3 Conclusions and future work

6.3.1 Contribution of the study

The aim of the study was to investigate whether ASDMs are suitable for the development of data warehouses. This was done by investigating literature on ASDMs followed by a literature study on data warehousing where Inmon and Kimball's data warehouse development approaches were investigated. To determine the suitability of ASDMs towards data warehouse development, a two-phased method was used.

During the first phase (reported in chapter 4) theoretical deductions were made on the suitability of ASDMs towards data warehouse development. The suitable and unsuitable theoretical characteristics of each ASDM were explained by examining in which phases of data warehouse development a certain ASDM's areas, properties, practices, principles or development phases could be applied to develop a data warehouse in a changing environment.

During the second phase the theoretical deductions made in chapter 4 were practically tested by conducting an interpretive experiment where each team used their assigned ASDM to guide their activities during the data warehouse development process. All teams followed an incremental and iterative development approach. Every increment or sub-data mart was developed and delivered incrementally to create the data mart. Where after the data mart, including everything from collecting requirements to GUI development and

report generation, was deployed as a whole to the users.

After examining the theoretical deductions made in chapter 4 and the interpretative experiment results in chapter 5, these deductions and interpretive results (propositions) were combined in chapter 6.

Since the resulting data warehouse was successful and the teams were able to follow their allocated ASDMs, it can be concluded that ASDMs are indeed suitable for data warehouse development according to Kimball's approach. However, it should be noted that the deductions made in chapter 4 limited the suitability of ASDMs to Kimball's approach.

6.3.2 Limitations

Two aspects of the study proved to limit the generalization of the results:

- Single data mart development: Better results on the incremental nature of data warehouse development and deployment could have been obtained if teams developed more than one data mart.
- Academic setting: The time spent by the teams on the project was limited, since the project formed part of a larger curriculum. Another factor that distinguishes this academic setting from an organisational setting is that participants did not receive any monetary award for their work.

6.3.3 Future research

This study has successfully indicated the suitability of ASDMs for data warehouse development. The results explained in this chapter are research findings that were gained from the deductions made in chapter 4 and the propositions explained in chapter 5. These findings could be presented as guidelines to develop a data warehouse using ASDMs in a constantly changing environment. Using these guidelines, data warehouses could be developed incrementally in an agile environment using one or a combination of suitable

ASDMs, where primary requirements are collected before development takes place.

REFERENCES

ABRAHAMSSON, P., SALO, O., RONKAINEN, J. & WARSTA, J. 2002. Agile software development methods: Review and analysis. Espoo 2002. VTT Publications. University of Oulu http://www2.umassd.edu/SWPI/xp/papers/p478.pdf Date of access: 14 Apr. 2005.

Agile Alliance. 2006. Helping Agile Projects Start & Helping Agile Teams Perform. http://www.agilealliance.org/ Date of Access: 9 Mrt. 2006.

AMBLER, S. 2002. Agile development best dealt with in small groups. *Computing Canada*: 9. 26 Apr.

ANDERSON, D.J. 2004. Feature-Driven Development: towards a TOC, Lean and Six Sigma solution for software engineering. Microsoft Corporation, Oct. 2004. http://www.agilemanagement.net/Articles/Papers/Feature_Driven_Developmen _-_towards_a_TOC_Lean_Six_Sigma_solution_v1_0.pdf Date of access: 7 Apr. 2006.

AVELING, B. 2004. XP Lite Considered Harmful? (*In* Eckstein, J & Baumeister, H., *eds*. XP 2004, LNCS 3092. Singer-Verslag Berlin Heidelberg 2004. p. 94-103.)

AVISON, D. & FITZGERALD, G. 2003. Information Systems Development Methodologies, Techniques and Tools. 3rd ed. England: Berkshire: McGraw-Hill. 592p.

AVISON, D. & FITZGERALD, G. 2003. Where now for development methodologies? *Communications of the ACM*, 46(1):79-82, Jan.

BOEHM, B. 2002. Get Ready for Agile Methods, with Care. *IEEE Computer*,

35(1):64-69.

BRINKKEMPER, S. 1996. Method Engineering: Engineering of information system development methods and tools. *Information and Software Technology*, (38):275-280.

CHIN, G. 2003. Agile project management: How to succeed in the face of changing project requirements. http://www.powells.com/cgi-bin/biblio?inkey=62-0814471765-0 Date of access: 23 Feb. 2006.

COCKBURN, A. 2001. Crystal Light Methods. *Cutter IT Journal.* http://alistair.cockburn.us/crystal/articles/clm/crystallightmethods.htm Date of access: 27 Mar. 2005.

COCKBURN, A. 2005. Crystal Clear: A Human-Powered Methodology for Small Teams. 1st ed. Addison-Wesley. 336p.

COCKBURN, A., & HIGHSMITH, J. 2001b. Agile Software Development: The People Factor. *Computer*:131-133. Nov.

COCKBURN, A., HIGHSMITH, J. 2001a. Agile Software Development: The Business of Innovation. *IEEE Computer*:120-122, Sep.

COHEN, D., LINDVALL, M., COSTA, P. 2003. A DACS State of the Art Report: Agile Software Development. Fraunhofer Center for Experimental Software Engineering Maryland and the University of Maryland. http://fc-md.umd.edu/fcmd/papers/DACS-SOAR-AgileSoftwareDevelopment.pdf Date of access: 13 Mar. 2005.

COLLIER, K. & HIGHSMITH, J. 2004. Applying Agile Practices to Data Warehousing. *Cutter Consortium: The Cutter Edge*, Dec. 28. http://www.cutter.com/ research/2004/edge041228.html Date of Access: 14

Jul. 2006.

CONBOY, K. & FITZGERALD, B. 2004. Towards a Conceptual Framework of Agile Methods. (*In* Zannier, C. et al., *eds*. XP/Agile Universe 2004, LNCS 3134. Springer-Verlag Berlin Heidelberg. p. 105-116.)

CONTROL CHAOS . 2006. What is Scrum? http://www.controlchaos.com/about/?SID=8ef7eb5b2a069a2710abef27d02c851f&SID=7da824062baf60b8e78ec5f99836f092 Date of access: 23 Feb 2006.

Control Chaos. 2005. What is Scrum? http://www.controlchaos.com/about/ Date of access: 23 Mar. 2005.

COPELAND, L. 2001. Extreme Programming. *Computerworld*, 3 Des. http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,66192,00.html Date of access: 2 Apr. 2006.

DSDM Consortium. 2003. DSDM and Data Warehousing: Synopsis. http://www.dsdm.org/kss/details.asp?fileid=92 Date of access: 28 Jun. 2006.

DE LUCA, J. 2005. Feature Driven Development Overview. Nebulon Pty.Ltd. http://www.nebulon.com/articles/fdd/download/fddoverview.pdf Date of access: 3 Mar. 2006.

DSDM Consortium. 2005. The History of the DSDM Consortium. http://www.dsdm.org/en/about/history.asp Date of access: 29 Mar. 2005.

FOWLER, F. & HIGHSMITH, J. 2001. The Agile Manifesto. *SDmagazine.com*, Aug. http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm Date of access: 9 Mar. 2006.

FOWLER, M. 2001. Put Your Process on a Diet. *Dr. Dobb's Portal, The World*

for Software Development. 29 Jun. www.sdmagazine.com/documents/ s=737/sdm0012a/ 0012a.htm Date of access: 19 Apr. 2005.

FOWLER, M. 2006. The New Methodology. http://www.martinfowler.com/ articles/newMethodology.html Date of access: 4 Mar 2006.

FRIEDMAN, J.H. 1998. Data Mining and Statistics: What is the Connection? http://www.salford-systems.com/doc/dm-stat.pdf Date of access: 10 Jul 2006.

GACHET, A. 2000. A Framework for Developing Distributed Cooperative Decision Support Systems – Inception Phase. University of Fribourg, Switzerland. http://www.gachet.net/alexandre/cv.html Date of access: 26 May 2006.

GACHET, A. & HAETTENSCHWILER, P. 2003. A De-centralized Approach to Distributed Decision Support Systems. *Journal of Decision Systems*, 12(2):141-158.

GOEDE, R. 2005. A Framework for the explicit use of specific systems thinking methodologies in data-driven decision support system development. Pretoria: UPE. (Theses – D.Phil.) 273 p.

GOOD, J.M. 2003. A Pragmatic Approach to the Implementation of Agile Software Development Methodologies in Plan-Driven Organisations. Lincoln University (Dissertation – Degree of Bachelor of Applied Computing with Honnours) 84p.

Government Technology. 2001. Business Intelligence: How Agencies can Breathe New Life into Old Data. A Government Technology Solution Spotlight: Microsoft. http://www.govtech.net/govcenter/solcenter/pdfs/GTMicrosoft10_ 01.pdf Date of access: 13 May 2006.

GRAZIANO, K. 2005. Agile methods and data warehousing. http://download-east.oracle.com/oowsf2005/628wp.pdf Date of access: 23 Feb 2006.

GREINER, L. 2001. Business Intelligence: Know what you do, do what you know. *Computer Canada*: 13-15, Mar.

HARDY, C.J., THOMPSON, J.B. & EDWARDS, H.M. 1995. The use, limitations and customization of structured systems development in the United Kingdom. *Information and Systems Technology*, 37(9):467-477.

HIGHSMITH, J. 2000. Retiring Lifecycle Dinosaurs. *Software Testing & Quality Engineering*: 22-28, Jul/Aug.

HIGHSMITH, J. 2002a. What is Agile Software Development? *CrossTalk The Journal of Defence Software Engineering*: 4-9, Oct.

HIGHSMITH, J. 2002b. Extreme programming: Agile project management advisory white paper service. http://rockfish.cs.unc.edu/COMP290-agile/xp-highsmith.pdf Date of Access: 23 Feb 2006.

HIGHSMITH, J. 2002c. Agile Software Development Ecosystems. 1st ed. Boston, MA.: Addison-Wesley. 448p.

HIGHSMITH, J. & COCKBURN, A. 2001. Agile Software Development: The Business of Innovation. *IEEE Computer*:120-122, Sep.

HISLOP, G.W., LUTZ, M.J., NAVEDA, J.F., McCRACKEN, W.M., MEAD, N.R. & WILLIAMS, L.A. 2002. Integrating Agile Practices into Software Engineering Courses. *Computer Science Education*, 12(3):169-185.

HOU, J., SHERMAN, C., O'BREIN, T. 1998. Data Warehousing on HP3000 Using IMAGE/SQL – A New Alternative. HPWORLD '98 Paper #2264.

http://datawarehouse.ittoolbox.com/pub/DWH3000.pdf Date of access: 26 May 2006.

HUIJBERS, R., LEMMENS, F., SENDERS, B., SIMONS, S., SPAAN, B., VAN TILBURG, P. & VOSSEN. 2004. Software Project Management: Methodologies & Techniques. (Software Engineering Project delivered as part of a study at Technische Universiteit Eindhoven in Department of Mathematics & Computer Science on 17 Sep. 2004.) Eindhoven. 37p. (Unpublished.)

HUISMAN, H.M. & IIVARI, J. 2006. Deployment of systems development methodologies: Perceptual congruence between IS managers and system developers. *Information & Management*, (43):29-49.

IIVARI, J. & MAANSAARI, J. 1998. The usage of systems development methods: are we stuck to old practices? *Information and Software Technology*, (40):501-510, 23 Jun.

IIVARI, J., HIRSCHEIM, R. & KLEIN, H.K. 1998. A paradigmatic analysis contrasting information system development approaches and methodologies. *Information System Research*, 9(2):164-193.

IIVARI, J., HIRSCHEIM, R. & KLEIN, H.K. 1999. Beyond Methodologies: Keeping up with Information Systems Development Approaches through Dynamic Classification. (*In* Proceedings of the 32nd Hawaii International Conference on System Sciences: 1-10.)

INMON, B. 1995. The Operational Data Store. *InfoDB*, 9(1):21-24, Feb. Available: Google Scholar. Date of access: 28 May 2006.

INMON, W.H. 1996. Building the Data Warehouse. 2nd ed. New York, N.Y.: Wiley. 401p.

INMON, W.H. 2000. What is a Data Warehouse? http://www.business.aau.dk/oekostyr/file/What_is_a_Data_Warehouse.pdf Date of access: 26 May 2006.

JEFFRIES, R. 2001. What is Extreme Programming? http://www.xprogramming.com/xpmag/whatisxp.htm Date of access: 1 Apr. 2006.

KIMBALL, R., REEVES, L., ROSS, M. & THORNTHWAITE, W. 1998. The data warehouse lifecycle toolkit. New York: Wiley. 771p.

LAWYER, J. & CHOWDHURY, S. 2004. Best Practices in Data Warehousing to Support Business Initiatives and Needs. (*In* Proceedings of the 37th Hawaii International Conference on System Sciences: 1-9.)

LEE. A.S. 1999. Researching MIS (*In* Currie, W.L., Galliers. R., *eds*. Rethinking management information systems. Oxford: Oxford University Press. P7-27.)

LINDBERG, P. 2003. Lean Development http://tesugen.com/archives/03/01/leandevelopment Date of access: 11 Apr. 2006.

LINDSTROM, L. & JEFFRIES, R. 2004. Extreme Programming and Agile Software Development Methodologies. *Information Systems Management*: 41-52, Summer.

LINDVALL, M., BASILI, V., BOEHM, B., COSTA, P., KATHLEEN, D., SHULL, F., TESORIERO, R., WILLIAMS, L. & ZELKOWITZ, M. 2002. Empirical Findings in Agile Methods. (*In* Proceedings of Extreme Programming and Agile Methods – XP/Agile Universe 2002. p.197-207.)

MACLING, B. 2004. Bullish on business intelligence. *Computing Canada*:20, 11 Jun. Available: Academic Search Premier. Date of access: 10 May 2006.

MAHNIC, V & DRNOVSCEK, S. 2005. Agile Software Project Management with Scrum. University of Ljubljana, Faculty of Computer and Information Science. 6p. http://www.mc.manchester.ac.uk/eunis2005/medialibrary/papaers/paper_194.pdf Date of access: 13 Apr. 2006.

MAILVAGANAM, H. 2004. Design Methodologies of Kimball and Inmon... Plus a Third Way. http://www.dwreview.com/Articles/KimballInmon.html Date of access: 1 Jun. 2006.

MCGUIGAN, B. 2006. What is Business Intelligence? http://www.wisegeek.com/what-is-business-intleeigence.htm Date of access: 15 May 2006.

MCKNIGHT, W. 2005. Why a Data Warehouse? http://www.datahabitat.com/pdf/ datawarehouse/.pdf Date of access:26 May 2006.

MENDONCA, J. 2002. The case for a less methodical methodology: lean, light, extreme, adaptive, agile and appropriate software development. (*In* Issues and Trends of Information Technology Management in Contemporary Organisations. 2002 Information Resources Management Association International Conference. Hershey, PA.: Idea Group Publishing. p.503-505.)

MEYER, D. & CANNON, C. Building a better data warehouse. 1ST ed. Upper Saddle River: Prentice Hall. 227p.

NORTON, D. 2005. Lean Software Development Overview http://codebetter.com/ blogs/darrell.norton/articles/50341.aspx Date of access: 11 Apr 2006.

PALMER, S.R. & FELSING, J.M. 2002. A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ.: Prentice-Hall. 271p.

PALVIA, P. & NOSEK, J.T. 1993. A field exanimation of system lifecycle techniques and methodologies. *Information & Management*, (25):73-84.

POPPENDIECK, M. 2003. Lean Software Development. C++ Magazine: Methodology Issue. Fall. http://www.poppendieck.com/pdfs/Lean_Software_ Development.pdf Date of access: 4 Apr. 2006.

QUARLES, D. 2002. Business Intelligence: The umbrella term. (*In* BWI-Werkstuk, Universiteit Amsterdam, Nov. 2002. http://www.few.vu.nl/ stagebureau/werkstuk/werkstukken/werkstuk-quarles.doc Date of access: 20 Aug. 2005.)

RAHM, E. & DO. H.H. 2000. Data cleaning: Problems and Current Approaches. University of Heipzig, Germany. http://www.cs.utah.edu/juliana/AdvancedDB/ Refs/data-cleaning-rahm-2000.pdf Date of access: 8 Sep. 2005.

REIFER, D.J. 2002. How Good are Agile Methods? *IEEE Software*: 16-18. Jul./Aug.

SAEKI, M. 1998. A meta-model for method integration. *Information and Software Technology*, (39):925-932.

SCHACH, S.R. 1997. Software Engineering with Java. USA: Irwin, R.D. 618p.

SCHWABER, K. & BEEDLE, M. 2002. Agile Software Development with Scrum. Upper Saddle River, NJ.: Prentice Hall. 158p.

SEAMAN, C.B. 1999. Qualitative Methods in Empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 425(4):557-572, Jul/Aug.

STEINDL, C. 2004. Lean Software Development. IBM Global Services, Application management Services. IT Architect and IT Specialist Institute Central Region 2004 in Herrenberg, Germany. http://agilealliance.org/articles/steindlchristophleans/file Date of access: 3 Apr. 2006.

STRAUBHAAR, J. & LA ROSE, R. 2003. Media now: Understanding media, culture & technology. 4th ed. Belmont, Calif.: Thomson/Wadsworth. 524p.

TechTarget. 2005. Business Intelligence. http://searchdatamanagement.techtarget.com/sDefinition/0,290660,sid91_gci213571,00.html

TDWI (The Data Warehousing Institute). 2004. TDWI Business Intelligence Fundamentals: From Data Warehousing to Business Impact. http://www.tdwi.org/files/pub/tdwi/BI_Fundamentals_Preview_1.pdf Date of access: 10 Mar. 2006.

WELLS, J.D. 2000. Extreme Programming Project. http://www.extremeprogramming.org/map/project.html Date of access: 2 Apr. 2006.

Wikipedia. 2006. Business Intelligence. http://en-wikipedia.org/wiki/Business_intelligence Date of access: 15 May 2006.

WINDLEY, P.J. 2003. Being smart about business intelligence. *InfoWorld*, 25(8):44, 24 Feb. Available: Academic Search Premier. Date of access: 12 May 2006.

WYNEKOOP, J.L. & RUSSO, N.L. 1993. Systems development methodologies: unanswered questions and the research-practice gap. (*In* DeGross, J.I et al., *eds*. Proceedings of the Fourteenth International Conference of Information Systems, Orlando, FL, 1993. p.181-190.

WYNEKOOP, J.L. & RUSSO, N.L. 1997. Studying systems development methodologies: an examination of research h methods. *Information Systems Journal*, (7):47-65.