A comparison of maintenance and support challenges within a data warehousing environment to that of a transactional application environment in a South African context

SM Juggath

21021430

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

Supervisor: Prof. Roelien Goede

October 2014

It all starts here ™

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT ®

Preface and Acknowledgement

This study on challenges, has ironically been a great challenge in my life. I could not have completed this study without the help, guidance and motivation from the following persons:

Firstly, Prof. Roelien Goede, for her guidance, patience, support and encouragement.

Secondly, I am my grateful to participants of this study who volunteered their time.

Thirdly, I would like to acknowledge my parents, Kishore and Shakuntala Juggath for their words of advice and motivation. I also received support from my in-laws, for which I am grateful.

Lastly, and most importantly, I would like to thank my wife, Verena, for her endless love, understanding, patience, support and encouragement during the writing of this dissertation.

Declaration

I, Shakeel Mitra Juggath declare that

A comparison of maintenance and support challenges within a data warehousing environment to that of a transactional application environment in a South African context

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

Signature: _____

Date: _____

Abstract

In transactional systems development literature, maintenance is reported as being a phase in the software development life cycle. In practice, this phase is often neglected as it occurs post-deployment and other ongoing projects take a higher priority. In data warehouse (DW) systems development literature, maintenance is not reported as being a phase but an ongoing iteration to the DW development project. It should therefore not be treated as a phase by DW systems professionals.

Although there is this fundamental difference in the approach to maintenance, transaction systems maintenance and DW maintenance share many of the same challenges. DW literature and methodologies inherently contain utilities and methods to assist in alleviating these challenges in a DW system. Transactional systems do not deal with these challenges inherently.

Research aspects were extracted from the literature review conducted. The literature review conducted demonstrates what the challenges in maintenance are, how the challenges of transactional systems compare to the challenges of DW maintenance and how the utilities and methods used in DW methodologies can inherently assist in managing these challenges from DW perspective. These research aspects were used to formulate an interpretive questionnaire.

This research portion of the study explores the use of DW systems development and maintenance methodologies in the industry among DW professionals. This is done by conducting an interpretive study using the interpretive questionnaire developed from the literature review. The interpretive questionnaire focusses on maintenance and dealing with the challenges thereof. Many themes evolved from the analysis of the interpretive study by using the content analysis method.

The final conclusions of this study is drawn by comparing and combining the information gathered from the literature review with the information gathered from the

interpretive study. Gaps are identified between practice and literature and recommendations are made based on these gaps.

Keywords: data warehouse maintenance, challenges in maintenance, data warehouse methodology, transactional system methodology, methodology comparison

Uittreksel

In literatuur oor die ontwikkeling van transaksiestelsels, word onderhoud as ’n aparte fase in die programmatuur se lewensiklus bespreek. In praktyk word hierdie fase dikwels afgeskeep aangesien dit na die ontwikkeling voorkom en ander projekte dan voorkeur geniet. In stelselontwikkelingsliteratuur oor datapakhuise word onderhoud nie beskryf as ’n aparte fase nie maar eerder as ’n volgende siklus van die datapakhuisontwikkelingsprojek. Dit moet daarom ook nie as ’n aparte fase deur professionele ontwikkelaars van datapakhuisstelsels gesien word nie.

Alhoewel daar hierdie fundamentele verskil in die benadering tot onderhoud tussen transaksiestelselonderhoud en datapakhuisonderhoud is, kom baie van dieselfde uitdagings voor. Die verskil is egter dat datapakhuisliteratuuur en -metodologieë ingeboude metodes en funksies bevat om hierdie uitdagings te hanteer terwyl die meeste transaksiestelselmetodologieë nie hierdie uitdagings inherent aanspreek nie.

Aspekte waaroor navorsing gedoen moet word is bepaal vanuit die literatuurstudie wat onderneem is. Die literatuurstudie dui aan wat die uitdagings van datapakhuisonderhoud is, hoe die uitdagings van transaksie-stelsels vergelyk met die uitdagings van datapakhuisonderhoud en hoe funksies en metodes wat in datapakhuismetodologieë gebruik word, inherent help om hierdie uitdagings uit ‘n datapakhuisperspektief te bestuur. Hierdie navorsingsaspekte is gebruik om ’n interpretiewe vraelys te ontwikkel.

Die navorsingsdeel van die studie ondersoek die gebruik van datapakhuis-stelselsontwikkeling en onderhoudmetodologieë in die industrie deur professionele datapakhuisontwikkelaars. Dit word gedoen deur ’n interpretiewe gevallestudie met behulp van ’n interpretiewe vraelys wat uit die literatuurstudie ontwikkel is. Die interpretiewe vraelys fokus op onderhoud en die hantering van die uitdagings daarvan. Verskeie temas het ontwikkel uit die analise van die interpretiewe studie met behulp van die inhoudsanalisemetode.

Die finale afleidings van die studie is gemaak deur die inligting van die literatuurstudie en die inligting van die interpretiewe studie te vergelyk en te kombineer. Gapings tussen die praktyk en die literatuur is geïdentifiseer en aanbevelings gegrond op hierdie gapings is gemaak.

Sleutelwoorde: datapakhuisonderhoud, uitdagings in onderhoud, datapakhuismetodologie, transaksiestelselmetodologie, metodologie vergelyking

Contents

Preface and Acknowledgement ii

Declaration iii

Abstract iv

Uittreksel vi

Contents viii

List of Figures xiii

List of Tables xiii

List of Code Segments xiv

CHAPTER 1 1

INTRODUCTION 1

1.1 Introduction 1

1.2 Background 3
1.2.1 The challenges of maintenance of transactional systems 3
1.2.2 The challenges maintenance in a data warehousing environment 5

1.3 Motivation for research 8
1.3.1 Motivation based on comparing the time-frame of software development and data warehouse development 9
1.3.2 Motivation based on volume of research done 10
1.3.3 Motivation based on questionable understandings and perceptions 12

1.4 Research aims and objectives 14

1.5 Research methodology 15

1.6 Chapterization 17

CHAPTER 2 22

RESEARCH METHODOLOGY 22

2.1 Introduction 22

2.2 What is research? 23
2.2.1 This study 23

2.3 Research design 24

2.3.1 Qualitative, quantitative and mixed methods 25
2.3.2 This study 27

2.4 Research perspectives in qualitative research 28
2.4.1 Positivistic perspective 28
2.4.2 Interpretive perspective 29
2.4.3 Critical social research 30
2.4.4 This study 31

2.5 Research methodologies 32
2.5.1 Positivistic research methodology 32
2.5.2 Interpretive research methodology 33
2.5.3 Critical social research methodology 36
2.5.4 This study 37

2.6 Research methods 38
2.6.1 Positivistic research methods 38
2.6.2 Interpretive research methods 42
2.6.3 Critical social research methods 53
2.6.4 This research 55

2.7 Data collection 56
2.7.1 Interviews 57
2.7.2 Observations 58
2.7.3 This study 60

2.8 Conclusion 60

CHAPTER 3 64

TRANSACTIONAL SOFTWARE DEVELOPMENT AND MAINTENANCE 64

3.1 Introduction 64

3.2 Software development models 65
3.2.1 The waterfall model 65
3.2.2 Software prototyping 68
3.2.3 The spiral model 69
3.2.4 Object oriented programming 71
3.2.5 Agile methodologies: Scrum 74

3.3 Software maintenance 80
3.3.1 The software maintenance process in the software life cycle process 81
3.3.2 Challenges in software maintenance 87
3.3.3 Addressing maintenance challenges 90

3.4 Conclusion 94

CHAPTER 4 98

DATA WAREHOUSE DEVELOPMENT AND MAINTENANCE 98

4.1 Introduction 98

4.2 OLTP, data warehouses and OLAP 100

4.3 Data warehouse methodologies 101
4.3.1 Principles of Inmon's DW 2.0 102
4.3.2 The Kimball life cycle approach 115

4.4 Challenges in data warehouse maintenance 140
4.4.1 Addressing maintenance challenges 142

4.5 Conclusion 142

CHAPTER 5 146

COMPARISON OF CHALLENGES OF TRANSACTIONAL SYSTEM MAINTENANCE AND DATA WAREHOUSE MAINTENANCE 146

5.1 Introduction 146
5.1.1 Methodology challenges revisited 147

5.2 Maintenance methodology 148
5.2.1 Transactional system maintenance methodology challenges 148
5.2.2 Transactional system maintenance methodology challenges from a DW perspective 149
5.2.3 DW maintenance methodology challenges 150
5.2.4 Addressing DW maintenance methodology challenges 150
5.2.5 Empirical research questions for maintenance methodology comparison 151

5.3 Perfective and preventive maintenance 152
5.3.1 Transactional system preventive maintenance challenges 152
5.3.2 Transactional system preventive maintenance challenges from a DW perspective 153
5.3.3 DW preventive maintenance challenges 154
5.3.4 Addressing DW preventive maintenance challenges 155
5.3.5 Empirical research questions for preventive maintenance comparison 157

5.4 Corrective Maintenance 157
5.4.1 Transactional system corrective maintenance challenges 158
5.4.2 Transactional system corrective maintenance challenges from a DW perspective 158
5.4.3 DW corrective maintenance challenges 159
5.4.4 Addressing DW corrective maintenance challenges 159
5.4.5 Empirical research questions for corrective maintenance comparison 160

5.5 Adaptive maintenance 160
5.5.1 Transactional system adaptive maintenance challenges 160
5.5.2 Transactional system adaptive maintenance challenges from a DW perspective 161
5.5.3 DW adaptive maintenance challenges 162
5.5.4 Addressing DW adaptive maintenance challenges 163
5.5.5 Empirical research questions for adaptive maintenance comparison 164

5.6 Summary of DW and OLTP maintenance literature 164

5.7 Conclusion 168

CHAPTER 6 172

INTERPRETIVE RESEARCH CONDUCTED 172

6.1 Introduction 172

6.2 Summary of DW and OLTP maintenance literature 173

6.3 Interview development 173

6.4 Data collection: Process and participants 179
6.4.1 Interview process 179
6.4.2 Participant detail 182

6.5 Data analysis 183
6.5.1 Strategy 184
6.5.2 Step 1 – Prepare the data 186
6.5.3 Step 2 – Define the unit of analysis 186
6.5.4 Step 3 – Develop categories and a coding scheme 186
6.5.5 Step 4 – Testing the coding scheme on a sample of text 187
6.5.6 Step 5 – Code all the text 189
6.5.7 Step 6 – Assess the coding consistency 193

6.6 Step 7 - Conclusions from the data 198
6.6.1 Theme 1: Challenges experienced 199
6.6.2 Theme 2: Communication 203
6.6.3 Theme 3: Cyclical development 206
6.6.4 Theme 4: Dimensional models 207
6.6.5 Theme 5: Documentation 210
6.6.6 Theme 6: High volumes 213
6.6.7 Theme 7: Metadata 214
6.6.8 Theme 8: Methodology 215
6.6.9 Theme 9: OLTP comparison 218
6.6.10 Theme 10: Outsourcing 219
6.6.11 Theme 11: Preventive/Perfective maintenance 221
6.6.12 Theme 12: Training 226
6.6.13 Theme 13: Use of tools 229

6.7 Conclusion 229

CHAPTER 7 232

CONCLUSIONS AND RECOMMENDATIONS 232

7.1 Introduction 232

7.2 Research aims and objectives revisited 232

7.3 Research evaluation 235
7.3.1 The principle of hermeneutic circle 235
7.3.2 The principle of contextualization 235
7.3.3 The principle of interaction between the researchers and the subjects 236
7.3.4 The principle of abstraction and generalization 236
7.3.5 The principle of dialogical reasoning 237
7.3.6 The principle of multiple interpretations 237
7.3.7 The principle of suspicion 238

7.4 Research discussion 238
7.4.1 Theme 1: Challenges experienced 238

7.4.2 Theme 2: Communication 240
7.4.3 Theme 3: Cyclical development 242
7.4.4 Theme 4: Dimensional models 243
7.4.5 Theme 5: Documentation 245
7.4.6 Theme 6: High volumes 246
7.4.7 Theme 7: Metadata 248
7.4.8 Theme 8: Methodology 249
7.4.9 Theme 9: OLTP comparison 249
7.4.10 Theme 10: Outsourcing 251
7.4.11 Theme 11: Preventive/perfective maintenance 252
7.4.12 Theme 12: Training 253
7.4.13 Theme 13: Use of tools 254

7.5 Final conclusions of study 255

7.6 Limitations of study 257

7.7 Further study considerations 258

7.8 Chapter summary 259

REFERENCES 260

APPENDIX A 277

INTERVIEW QUESTIONNAIRE 277

Interview introduction 277

Interview Questions 278

APPENDIX B 281

CODE LIST 281

APPENDIX C 289

LINKS BETWEEN CODES 289

APPENDIX D 308

CODE FAMILIES 308

List of Figures

FIGURE 1 - 1 CHAPTER ORGANISATION 18
FIGURE 2 - 2 PERSPECTIVES OF QUALITATIVE RESEARCH (MYERS, 1997:242) 28
FIGURE 3 - 1 A TYPICAL WATERFALL PROCESS (B. BOEHM, 2006: 15; BRAUDE & BERNSTEIN, 2011: 37) 66
FIGURE 3 - 2 THE SPIRAL MODEL (PRESSMAN, 2001: 37) 70
FIGURE 3 - 3 ACTIVITIES IN THE MAINTENANCE PROCESS (ISO/IEC 14764, 2006: 6) 82
FIGURE 4 - 1 "THE KIMBALL LIFE CYCLE DIAGRAM" (KIMBALL *ET AL., 2*008: 3) 116
FIGURE 4 - 2 A RETAIL SALES PROCESS DIMENSIONAL MODEL (KIMBALL *ET AL., 1*998: 145) 126
FIGURE 6 - 1 CODE HIERARCHY FROM ATLAS.TI 196

List of Tables

TABLE 1 - 1 COMPARISON OF NUMBER OF RESULTS FOR SEARCH TERMS SOFTWARE MAINTENANCE, DATA WAREHOUSE MAINTENANCE, BUSINESS INTELLIGENCE MAINTENANCE FROM SCHOLAR SEARCH PROVIDER GOOGLE SCHOLAR AS WELL AS SCHOLAR PUBLISHERS IEEE AND ACM. 11
TABLE 2 - 1 SUMMARY OF PRINCIPLES FOR INTERPRETIVE FIELD RESEARCH (KLEIN & MYERS, 1999: 72) 34
TABLE 3-1: TEMPLATE FOR A SPRINT TASK BOARD ("TRAINING FOR SCRUM TASK BOARD USE," 2011; KNIBERG, 2007: 53) 78
TABLE 4 - 1 SUMMARY OF DATA SECTORS IN DW 2.0 109
TABLE 4 - 2 KIMBALL LIFE CYCLE ROLES AND MEMBERS SUMMARISED FROM KIMBALL *ET AL.* (2008: 33) 119
TABLE 4 - 3 DIFFERENCES BETWEEN PROGRAM AND PROJECT REQUIREMENTS DEFINITION EFFORTS (KIMBALL *ET AL., 2*008: 65) 122
TABLE 4 - 4 EXAMPLE OF A BUS MATRIX FOR A MANUFACTURING SUPPLY CHAIN (KIMBALL *ET AL., 2*008: 251) 127
TABLE 5-1 SUMMARY OF EMPIRICAL RESEARCH AREAS 167
TABLE 6-1: EMPIRICAL RESEARCH ASPECTS 173
TABLE 6-2: INTERVIEW QUESTIONS, AIMS AND REFERENCE 179
TABLE 6-3 PARTICIPANTS 182
TABLE 6-4 ZHANG AND WILDEMUTH'S (2009: 3) 8 STEP PROCESS TO PERFORMING CONTENT ANALYSIS 185
TABLE 6-5 UNDIRECTED CODE COUNT FOR FIRST FOUR RESPONSES 191
TABLE 6-6 CODES THAT WERE NEWLY CREATED PER DOCUMENT 195
TABLE 6-7 COUNT OF CODES PER DOCUMENT FOR FAMILY CHALLENGES EXPERIENCED 200
TABLE 6-8 COUNT OF CODES PER DOCUMENT FOR FAMILY COMMUNICATION 203
TABLE 6-9 COUNT OF CODES PER DOCUMENT FOR CYCLICAL DEVELOPMENT 207
TABLE 6-10 COUNT OF CODES PER DOCUMENT FOR DIMENSIONAL MODELLING 208
TABLE 6-11 COUNT OF CODES PER DOCUMENT FOR DIMENSIONAL MODELLING 210
TABLE 6-12 COUNT OF CODES PER DOCUMENT FOR HIGH VOLUMES 214
TABLE 6-13 COUNT OF CODES PER DOCUMENT FOR METADATA 214
TABLE 6-14 COUNT OF CODES PER DOCUMENT FOR METHODOLOGY 216
TABLE 6-15 COUNT OF CODES PER DOCUMENT FOR OLTP COMPARISON 219
TABLE 6-16 COUNT OF CODES PER DOCUMENT FOR OUTSOURCING 219
TABLE 6-17 COUNT OF CODES PER DOCUMENT FOR PREVENTIVE/PERFECTIVE MAINTENANCE 222
TABLE 6-18 COUNT OF CODES PER DOCUMENT FOR TRAINING 226
TABLE 6-19 COUNT OF CODES PER DOCUMENT FOR USE OF TOOLS 229
TABLE 7-1 RESEARCH ASPECTS RELATED TO CHALLENGES EXPERIENCED 240
TABLE 7-2 PARTICIPANT RESPONSE INDICATING SEPARATION FROM USERS 241
TABLE 7-3 RESEARCH ASPECTS RELATED TO CODING THEME "COMMUNICATION" 242
TABLE 7-4 RESEARCH ASPECTS RELATED TO CODING THEME "DIMENSIONAL MODELS" 244
TABLE 7-5 RESEARCH ASPECTS RELATED TO CODING THEME "DOCUMENTATION" 246
TABLE 7-6 RESEARCH ASPECTS RELATED TO CODING THEME "HIGH VOLUMES" 247
TABLE 7-7 RESEARCH ASPECTS RELATED TO CODING THEME "OUTSOURCING" 251
TABLE 7-8 RESEARCH ASPECTS RELATED TO CODING THEME "PREVENTIVE/PERFECTIVE MAINTENANCE" 252
TABLE 7-9 RESEARCH ASPECTS RELATED TO CODING THEME "TRAINING" 253

List of Code Segments

CODING EXCERPT 1: P2 RESPONSE TO CHALLENGES OF DW MAINTENANCE 180
CODING EXCERPT 2: P3 RESPONSE TO CHALLENGES OF DW MAINTENANCE 181
CODING EXCERPT 3: CODES FOR P2 'S AND P3'S RESPONSE TO CHALLENGES OF DW MAINTENANCE 187
CODING EXCERPT 4: DIFFERENCES BETWEEN TELEPHONIC INTERVIEW AND WRITTEN INTERPRETIVE QUESTIONNAIRE FOR CODE "CYCLICAL DEVELOPMENT NOT BEING PRACTISED" 191
CODE EXCERPT 5 - EXAMPLE OF CODE NEIGHBOUR LIST 197
CODE EXCERPT 6 – EXTRACT FROM "ALL CODES WITH QUOTATIONS REPORT" 197
CODE EXCERPT 7 – HIGH VOLUMES POSE CHALLENGE 200
CODE EXCERPT 8– COMMUNICATION CHALLENGES OF OUTSOURCING 201
CODE EXCERPT 9 – COMMUNICATION CHALLENGES OF OUTSOURCING 204
CODE EXCERPT 10 – STEERING COMMITTEE NOT BEING UTILISED 205
CODE EXCERPT 11 – MATERIALIZED VIEW PROBLEM 208
CODE EXCERPT 12 – DOCUMENTATION USEFULNESS NEGATIVE 211
CODE EXCERPT 13 – LENGTHY DOCUMENTATION IS COUNTER PRODUCTIVE 212
CODE EXCERPT 14 – METADATA NOT UNDERSTOOD BY DEVELOPER 215
CODE EXCERPT 15 – INMON TYPE OF PHILOSOPHY 216
CODE EXCERPT 16 – TECHNOLOGY RELIANCE AND DEPENDENCY 217
CODE EXCERPT 17 – DIFFERENT DEVELOPERS DOING MAINTENANCE 220
CODE EXCERPT 18 – OUTSOURCED DEVELOPERS HAVE TO SPEND TIME LEARNING 220
CODE EXCERPT 19 – OUTSOURCING NEGATIVE 221
CODE EXCERPT 20 – LOW PRIORITY ON PERFECTIVE MAINTENANCE 223
CODE EXCERPT 21 – HARDWARE IMPROVEMENT TO COPE WITH VOLUME 225
CODE EXCERPT 22 – HARDWARE IMPROVEMENT TO COPE WITH VOLUME 227
CODE EXCERPT 23 – DW PROFESSIONAL TRAINING 227

Chapter 1
Introduction

1.1 Introduction

The purpose of this chapter is to introduce the topic of research, along with “why” and “how” for the research. The “why” portion will give motivation for the research and the “how” portion will explain the research methodology briefly.

This research is concerned with comparing the challenges of maintenance of data warehouses and business intelligence systems with the challenges of maintenance and support of transactional software systems. The relevant terms regarding this research topic are defined below.

Kimball *et al.* (1998: 19) define a data warehouse simply as “the queryable source of data in the enterprise”. Inmon *et al.* (2008: 7) define a data warehouse as being: “subject oriented, integrated, non-volatile, time variant, a collection of data in support of management’s decision”

The terms “data warehousing” and “data warehouse” (DW) have become synonymous and ambiguous with business intelligence (BI) as has the term DW team/developer to BI team/developer. Kimball *et al.* (2008: 10) addresses this ambiguity by referring to “the queryable data in your DW/BI system as the *enterprise data warehouse,* and value-add analytics as *BI applications.”* In scholarly articles, the more common term is data warehousing therefore the phrase used in this dissertation will be data warehouse unless business intelligence is the term being used in the source being cited.

Similarly, the differences in the phrases software engineering and software development are unclear. These terms will be used interchangeably in this dissertation, usually according to the source being referred to at that point in the text.

The main focus of our study is software maintenance. Bennet and Rajlich (2000: 1) have quoted the ISO 1995 definition of software maintenance as “the software

product undergoes modification to code and associated documentation due to a problem or the need for improvement. The objective is to modify the existing software product while preserving its integrity."

IEEE Standard 1219 defines software maintenance as "Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment." (IEEE Std 1219, 1993: 4)

The terms maintenance and support are often grouped together. Kitchenham *et al.* (1999: 378) summarise the support levels provided by maintenance organisations into three levels:

"
- Level 1—the help desk staff are non-technical, and are responsible for logging problems and identifying the technical support person most likely to be able to assist a user.
- Level 2—the technical support personnel know how to communicate with users and understand their problems, and they can advise on workarounds and quick fixes.
- Level 3—the maintenance engineers are authorised to make changes to the product."

From the above definitions, it can be stated that the term maintenance refers to the modification of software whereas support generally refers to communication with users. In this dissertation, the term maintenance includes the support components which are encompassed in these 3 levels of Kitchenham *et al.*'s summary.

Background on software maintenance is discussed in Section 1.2. This is done in the separate context of transactional software systems and data warehouses in Sections 1.2.1 and 1.2.2 respectively. These sections are based on both experience gained from working in a support role and on available literature on the topic.

Section 1.3 goes into detail of the motivation for the study. The section discusses the amount of literature available on the topic of maintenance and support of data warehouse systems and does a comparison with the amount of literature available for transactional software maintenance. The motivation for the study is followed by the aims and objectives of the study in Section 1.4

The research methodology is discussed in Section 1.5. This section is a preliminary discussion of the choice of research methodology which is discussed further in Chapter 2.

Section 1.6 discusses the limitations of the study. This chapter concludes with a summary of chapters to follow.

1.2 Background

The background presents the challenges experienced in support and maintenance of transactional systems and that of data warehousing systems.

1.2.1 The challenges of maintenance of transactional systems

Software development has a cost factor for any business. Whether the development is in-house or out-sourced, time is required from human resources and there are many other expenses in the forms of acquisition of hardware, software licensing and training of users. Software development usually takes the form of a project which generally has an end-date that coincides with deployment. After deployment of a transactional system, most of the development team are rolled off the project whilst a much smaller maintenance and support team remain behind (Kimball *et al., 2*008: 580). The deployment signals a milestone where executives begin to expect the tapering of costs.

If the project was executed well and care and attention was paid to the testing phase (where any bugs were reported and fixed), this expectation of tapering costs is a natural one. Unfortunately, costs do not always taper as rapidly as one would expect. Modern systems are rarely standalone systems. As the need to interact with other

systems increase, so too does the requirement for change and maintenance (Manchanda *et al., 2*009: 114; Mookerjee, 2005: 76). Therefore, expectations of tapering costs are often unreasonable. This unreasonable expectation is proliferated by the attitudes of managers that "maintenance is a waste of time" and further proliferated by the attitudes of developers that "maintenance is difficult and boring" (Stachour & Collier-Brown, 2009: 54).

What makes developers feel that maintenance is difficult and what makes managers feel as if maintenance is a waste of time? Anquetil *et al.* (2007: 515) put forward that maintenance developers have to have a much broader knowledge of the software system spanning "application domain, system's architecture, particular algorithms used, past and new requirements, programming language, development environment, etc." As gaining this knowledge requires detailed analysis of the system, it is costly and time-consuming to achieve (Anquetil *et al., 2*007: 515). It follows that without even making any changes or doing any coding, much time is spent merely understanding the system that needs to be maintained. This can be frustrating for managers who feel that the developer is not being productive and developers who might feel that their work is not being appreciated.

Other than perceptions of managers and the "knowledge management challenge" (Anquetil *et al.,* (2007: 515) of learning a system's ins and outs, there are factors which influence maintenance time and its allocation. Dekleva (1992: 359) included the following factors as possible influences on maintenance time allocation: "system size, number of users, system class, system age, maintenance organisation, use of software tools, ability of maintainers, organizational stability".

Part of the conclusions drawn by Dekleva (1992: 359) was that system size does influence maintenance costs as larger systems tend to be more complex which leads to losses in productivity and increased maintenance. An increase in the number of users also increases the amount of maintenance required. Maintenance of a system is generally high in the first two years of its use and thereafter maintenance should be minimal, however some methodologies do encourage mandatory changes and

functional enhancements which could affect the time taken on maintenance after the initial two year period.

In line with Dekleva's (1992: 359) findings regarding maintenance being more difficult on larger systems is Mookerjee's (2005: 77) discussion on maintenance costs. Mookerjee states maintenance costs are higher on systems which are co-dependent on other systems as a change to any of the interacting systems may result in changes to all systems. Mookerjee (2005: 77) also describes the waiting cost of maintenance as the possible revenue lost by a company system due to maintenance. The waiting cost in particular is a stressful one for maintenance developers as integral systems to the business would pose a high waiting cost for every minute that the system is non-functional. During unplanned maintenance, this would be very stressful and pose a key challenge to maintenance developers in troubleshooting problems.

In summary, the main challenges of maintenance from a management perspective is to reduce the time and cost of maintenance. From a development perspective, the main challenges are of a knowledge management nature. Some of factors that influence both management and developers are: the size of the systems, the co-dependency of the system, the number of users and the age of the system. These factors will be re-looked at in Chapter 3 when the software development life cycle is reviewed.

1.2.2 The challenges maintenance in a data warehousing environment

Data warehousing is becoming an essential part of business strategy development; which makes it a significant portion of a company's information technology solution (Negash, 2004: 185). Legodi and Barry (2008: 1) report a Gartner estimate that enterprise data warehousing and business intelligence will make up 12% of the overall information technology budget for most corporate companies.

Therefore, a successful business will need to focus on data warehousing. Somewhat ironically, Kimball *et al.* (2008: 2) state that successful data warehousing depends on focusing on the business; as well as dimensionally structuring data; and incremental development of the warehouse as opposed to a Big Bang approach.

Inmon *et al.* (2008: 8) differentiate between transactional system development and data warehouse development by saying, “developers around the world are used to gathering requirements and then building a system. This time-honoured approach is drummed into the heads of developers as they build operational systems. But a data warehouse is built quite differently. It is built iteratively, a step at a time. First one part is built, then another part is built, and so forth. In almost every case, it is a prescription for disaster to try to build a data warehouse all at once, in a “big bang” approach.”

To avoid this disaster, best practice for DW development as described by Kimball is that “on-going care and nurturing of the warehouse simply can't be turned over to a separate maintenance organization while the DW/BI team moves on to tackle the next implementation” (Kimball *et al., 2*008: 579). Kimball argues that a DW/BI system is never finished hence the original team who spent time gathering requirements by doing interviews, writing business cases and drawing up the specification documents have already gathered the knowledge necessary for the next round of development and should be the team to keep going with the “support and maintenance”. Kimball *et al.* (2008: 585) go on to stress that a good DW system will evolve and grow and will therefore always have requests for integration with other systems, better performance and additions to data mining module and key performance indicators (KPIs). It is these concepts of continuous improvement and a project that never quite ends which are the major misunderstandings of the data warehouse life cycle (DWLC) when it comes to support and maintenance.

As a result of this lack of understanding, there may be a deviation of the best practice approaches in the form of outsourcing. Outsourcing has become a growing trend in the 21st century as companies concentrate more and more on their core

competencies and key business (Salonen & Pirttimäki, 2005: 663). One benefit of outsourcing a BI function is the initial cost involved. As it may not be financially viable to hire a fully-fledged BI department, it makes more sense to outsource an initial BI venture to test how valuable BI could be (Salonen & Pirttimäki, 2005: 668).

However, Salonen and Pirttimäki (2005: 671) also warn that BI results are best when a BI unit is located physically close to end-users and end-users and BI developers are working as a team. Unless propagated by management, this type of team does not always work when outsourcing. They further warn that outsourcing BI to multiple vendors might seem a good idea as a single vendor may not be able to efficiently cater for all BI needs; however this could lead to miscommunication and incompatible intelligence products. Multiple BI units could also increase costs of the management and coordination of these units.

Outsourcing BI may also deviate from best practice in the sense that it could lead to knowledge management issues if the original outsourcing company is not kept on to continue with continuous iterations of the data warehouse implementation. As Kimball *et al.* (2008: 579) point out; the original DW development team should be the team to continue its development. If this does not happen, maintenance developers could have knowledge management issues such as lack of requirement specifications, lack of documentation, and busy users who are not keen to spend time repeating requirements and problems to another development team (Anquetil *et al., 2*007: 518).

Many information technology (IT) executives may follow a transactional software development life cycle approach to data warehouse maintenance by reallocating resources after deployment. “Unfortunately, many chief information officers (CIOs) presume that the size of their DW/BI team will halve when the new system goes into production. This is an unrealistic expectation...”(Kimball *et al., 2*008: 563). This expectation again stems from lack of understanding of DWLC. One reason for the requirement of keeping on resources after deployment is that users begin to realise what can be done with the data warehouse and then only do users understand what

the possibilities are. Therefore, after the first deployment there are more requirements than prior to the initiation of the project (Inmon *et al., 2*008: 9).

Inmon (2008: 9) states “the biggest failures in the building of a data warehouse occur when developers treat it as if it were just another operational application system to be developed.” This statement and the previous paragraph summarises the reasons and challenges for data warehouse development as well as maintenance. Developers with lack of understanding deploy the data warehouse and these developers then get re-allocated to other projects by executives and managers who also do not have a good understanding of the DWLC. This leaves behind a short-staffed maintenance team with all the challenges of a flood of new requirements that naturally follow an initial DW deployment.

The concerning issue of lack of understanding of the DWLC is further explored in Section 1.3.3 as it is part of the motivation for this study.

1.3 Motivation for research

It has thus far been established that the challenges of development and maintenance of data warehouses differ to those of transactional systems. The major difference when it comes to maintenance is that DW methodologies do not include a defined maintenance phase after initial deployment but rather a natural progress into the next iteration of development (Kimball *et al., 2*008: 579).

Having also established that data warehousing and software maintenance have important roles to play in information technology, it may be concluded that there is a requirement for research to be done in the area of data warehouse maintenance. However, before that conclusion is drawn, a literature review of the topic of data warehouse maintenance needs to be performed. This literature review is done in Chapter 5. For immediate purposes of demonstrating a motivation for the study, three major motivational factors have played a role: Section 1.3.1 examines the time-period of the advent of data warehousing in comparison to time-period to the advent software development. Section 1.3.2 examines the volume of research done on

these topics and Section 1.3.3 examines the perceptions and understanding of data warehouse methodologies by people involved with data warehousing.

1.3.1 Motivation based on comparing the time-frame of software development and data warehouse development

Research in the field of software development has been on-going for at least five decades. In 1966, there was study done by Asher Opler on the developments of software from 1960 to 1966. Opler (1966: 1763) summarised the challenges in during this 6 and a half year period as:

" 1) Increasing software requirements under pressure of user demand and strong hardware dependence on software.

2) Failure to solve production problems despite major emphasis on production and quality control.

3) Acceptance of the use of programming languages.

4) Attempts to control proliferation of different languages.

5) User acceptance of the dominance of large (software) supervisory systems.

6) Realization of the huge cost of transition to next-generation computers.

7) Unsuccessful attempts to develop completely satisfactory general techniques for automatic software production and general techniques for automatic translation of programs from one computer to another."

It is interesting to observe that many of the challenges experienced more than 40 years ago by computer scientists are still considered challenges today. (Voas, 2007: 48) lists 13 challenges of software engineering. Some of these include software quality where requirements and user demand are mentioned, return on investment which correlates to the realization of cost which Opler (1966) mentions, measurement and metrics which correlates to quality control in Opler's (1966) article, and process improvement which relates to point 7 (automated software production) in Opler's (1966) summary.

The most striking observation however, is that there was research done on this topic of software engineering 40 years ago and similar research is on-going today. Given this time span, it is fair to say that there has been a substantial amount of research produced in the topic of software engineering. A large part of software development is the maintenance phase. This phase is often considered the most costly phase of a software development life cycle (Ahmed, 2006: 450). As such, research on software maintenance has also been going on since the advent of software development.

Almost from the advent of studies into software development, there was interest in decision support systems (DSS). Around 1965, powerful mainframe systems made it feasible to develop management information systems (MIS) in large companies (Power, 2003). However it was only until the early 1990s did a major technology shift occur from mainframe-based DSS to client/server-based DSS. During this time Bill Inmon and Ralph Kimball actively promoted relational based database technologies and database management systems (DBMS) and published their books “Building the Data warehouse” (Inmon, 1992) and “The Data Warehouse Toolkit” (Kimball, 1996) respectively. These were influential in vendors recognising that decision support was different from online transactional processing (OLTP) and began implementing online analytical processing (OLAP) capabilities into their databases (Power, 2003).

It was therefore only after this major technology shift that real research began on the topic of data warehouses as we refer to them today. As can be seen, whilst software engineering as well as software maintenance has been a topic of academic interest since prior to 1960, relational based data warehouses and OLAP have only had similar interest since after 1990. As data warehousing is also a subset of software in general, it is conclusive that substantially less research been undertaken on the topic of data warehouse development and data warehouse maintenance than there has been of software development and software maintenance.

1.3.2 Motivation based on volume of research done

The Google Scholar engine examines the largest and most popular scholar publishers and university presses. A few of these are: IEEE, ACM, Macmillan and

Wiley (Jacsó, 2005: 209). Therefore, the number of results returned for different search terms would give one a fair indication comparative amount of publications which contain those terms. Table 1.1 does this comparison and shows the results of the different search terms from Google scholar as well as other popular scholar publishers as at the start of this study on 17 February 2011.

Search Option where words occur anywhere in the article	Search Term	Results – Google Scholar	Results – IEEE Xplore Digital Library	Results – ACM Digital Library
"with all of the words"	Software maintenance	about 1 970 000	98,595	23,703
	"Data warehouse" maintenance	about 20 100	5,234	671
	"Business Intelligence" maintenance	about 11 800	818	241
	Software support	about 2 340 000	453,396	97,128
	"Data warehouse" support	about 39 100	4 851	1,520
	"Business Intelligence" support	about 29 400	3,087	841
"with the exact phrase"	Software maintenance	about 49 400	13,751	5,538
	Data warehouse maintenance	about 339	59	43
	Business Intelligence maintenance	3	0	0
	Software support	about 37 600	5,209	1,452
	Data warehouse support	about 158	15	5
	Business Intelligence support	about 49	2	0

Table 1 - 1 Comparison of number of results for search terms software maintenance, data warehouse maintenance, business intelligence maintenance from scholar search provider Google Scholar as well as scholar publishers IEEE and ACM.

Table 1.1 shows that software maintenance and software support is an explored research topic. Software maintenance within data warehousing systems has not enjoyed as much academic attention. This does not infer that data warehouse maintenance does not warrant that attention. One might argue that data

warehousing maintenance and support would form a subset of software maintenance and support and hence it has not demanded special attention. However, the opposing side of this argument would be that even though data warehouse development forms a subset of software development, there is there a substantial amount of research conducted on the data warehouse life cycle (DWLC) methodology and there have also been many books published on the subject, most notably by authors Ralph Kimball, such as (Kimball, 1996; Kimball *et al., 1*998; Kimball *et al., 2*008) and William Inmon, such as (Inmon, 1992, 2002, 2005; Inmon *et al., 2*008). Therefore, there seems to be a lack of literature dedicated to data warehouse maintenance.

1.3.3 Motivation based on questionable understandings and perceptions

Ralph Kimball and William Inmon have been nicknamed “the doctor of DSS” and “the father of data warehousing” respectively (Power, 2003: 5). These authors have separately written of the iterative development approach for data warehousing as opposed to a “big bang” approach followed by a maintenance phase (Kimball *et al., 1*998; Inmon, 2002). In 2004, Negash (2004: 186) searched for BI curricula being taught at universities worldwide and only found 5 universities. This minute figure is likely due to the search term being “business intelligence” as opposed to “data warehousing” or “decision support systems” however, it does give one an idea of how fresh and unexplored the subject of data warehousing and business intelligence is relative to software development. This idea is further supported when viewing the results and discussion around the search term comparisons done in Table 1.1.

As data warehousing is a young discipline when compared to software development, a standardisation of the development process is yet to occur. Many data warehouse project failures may be attributed to the complexities of the development process (List *et al., 2*002: 204).

The data warehouse network, who have merged with the Business Intelligence Division of Sybase Inc. (“DATA WAREHOUSE NETWORK,” 1998) have reported

that 70% of data warehouses are a failure. This is based on a survey which had defined a data warehouse failure as occurring when the initial “design, scope, strategy, infrastructure or technology” is abandoned (Inmon, 2007). Inmon (2007) strongly disagrees with the results of their survey arguing that definition used for a failure is “worse than circumspect”. The reason given for this is that altering a data warehouse design and strategy is following the time honoured approach of iterative development and hence does not constitute a failure, but rather, it constitutes a success.

Inmon (2007) also argues against a second study in the same article. This was an organization and technology research (OTR) study for which the results were released in 1997. The study concluded that “the companies surveyed had yet to show financial benefits of a data warehouse”. On this point, Inmon argues that this conclusion was too premature as the companies surveyed had no real experience with data warehousing as data warehousing was still in infancy in 1997. Inmon further questions the conclusion stating that the study grouped data marts in with data warehouses which brings into doubt the understanding of those performing the survey.

From List’s (2002: 204) argument, it seems that there might be a lack of understanding by software professionals on how to implement data warehouses which causes failures in data warehouse projects. This lack of understanding of a complex development process would carry over into a maintenance phase especially when the maintenance phase is meant to be a new iteration of development.

Inmon’s (2007) argument identifies a possible lack of understanding by researchers as well as non-software professionals in companies that have data warehouses implemented of what constitutes a data warehouse failure. These non-software professionals are the data warehouse users and sponsors whose opinion of the data warehouse is a vital factor in whether or not it is a success (Shin, 2003: 146).

It therefore follows that the success factor of data warehouse maintenance and the challenges experienced by a data warehouse maintenance team are highly influenced by the opinions of the users and sponsors of the data warehouse. It also follows that these challenges and successes of a data warehouse maintenance team are also influenced by the understanding of the data warehouse development process by the implementers of the data warehouse.

But are these opinions (of users and sponsors) correct – especially given a South African context? And did the implementers within a South African industry understand the incremental process of building a data warehouse? As shown in Section 1.3.1, data warehousing is a new field of study and as shown in Section 1.3.2, there is substantially less research available on the topic of data warehousing than there is on software development. Bases on the researcher's experience in the industry, the inclination is that the answer to these questions are "no". However, to truly understand the challenges of data warehousing and the reasons for these challenges one feels that these are questions which require further study and investigation.

The investigation first explores defined methodologies for both transactional and data warehouse support and maintenance and then attempt to map the challenges experienced by maintenance developers to these methodologies in order to discover whether or not the challenges experienced by data warehouse maintenance and support developers are due to the deviation from data warehouse development methodologies to transactional software development methodologies.

1.4 Research aims and objectives

The main objective of this dissertation is to compare the challenges between maintenance of transactional systems and the maintenance of data warehousing systems.

From the interviews and surveys that have been conducted, one of the most important aims is to understand whether maintenance of data warehousing systems

in a South African context is being performed as recommended in data warehousing methodologies. In order to do this, the following sub-objectives are addressed:

1. To understand transactional software development and maintenance methodologies as well as to understand challenges of maintenance of transactional systems by doing a literature review.
2. To understand data warehouse development and maintenance methodologies as well as to understand challenges of maintenance of data warehouse systems by doing a literature review.
3. To compare the challenges of transactional software maintenance and the challenges of data warehouse maintenance by examining the literature reviews conducted.
4. To understand the data warehouse maintenance methodologies and maintenance challenges being practiced in South Africa and to understand whether these practices and the management of these challenges correlate to data warehousing methodologies.
5. To present the findings on the challenges of maintenance and support of a data warehousing environment compared the challenges of maintenance and support of transactional systems.

1.5 Research methodology

A choice was taken to follow a qualitative research approach. The motivation for qualitative research as Myers (1997: 241) puts it, “comes from the observation that, if there is one thing which distinguishes humans from the natural world, it is our ability to talk!” Further motivation is that the proposed type of research does not easily lend itself to being partitioned into discrete entities but rather lends itself to investigating the meaning and context of the type of methodologies used in data warehouse maintenance which closely mirrors the strengths of qualitative research methods described by Kaplan and Maxwell (2005a: 31).

Myers (1997:241) discusses three perspectives to qualitative research: positivist, interpretive and critical theory. A more detailed discussion on research approach and research perspectives is provided in Chapter 2. The research perspective

undertaken has been an interpretive one. “Interpretive studies assume that people create and associate their own subjective and intersubjective meanings as they interact with the world around them. Interpretive researchers thus attempt to understand phenomena through accessing the meanings that participants assign to them. ... interpretive studies reject the possibility of an "objective" or "factual" account of events and situations, seeking instead a relativistic, albeit shared, understanding of phenomena.” (Orlikowski & Baroudi, 1991: 5). The research performed assumes that maintenance data warehouse developers believe that they are following a correct data warehouse methodology and seeks to build an understanding of whether this belief is true.

While reviewing literature on transactional software maintenance methodology and data warehouse maintenance methodology, an understanding of methodologies and challenges of maintenance has been cultivated. Methods to address these challenges have also been investigated. In order to cultivate an understanding of whether data warehouse developers have an understanding of these methodologies, and whether they too experience similar maintenance challenges and have methods to address these challenges, interviews with maintenance data warehouse developers have be conducted. The nature of the interview has been semi-structured with a combination of open-ended questions and follow-up questions requiring the interviewee’s to share experiences and opinions in order to establish links to the methodologies and maintenance challenges in available literature. The information gathered is qualitative rather than quantitative.

Content analysis is a method that examines language to identify patterns in large amounts of text (Hsieh & Shannon, 2005: 1278). Content analysis is the method that has been followed under the interpretive research perspective and is used to analyse the data collected from interviews.

1.6 Chapterization

Figure 1-1 on the next page contains an organisational diagram of the dissertation. This diagram shall precede every chapter to indicate the position of the reader in the dissertation and remind the reader of the purpose of the chapter(s) to follow and also the purpose of the preceding chapter(s). The blocks at the top of Figure 1-1, shaded in black, titled “Chapter 1” on the left and “Introduction and Problem Statement” on the right indicate the current position in the organisational diagram. The diagram also illustrates how the chapters link together to achieve the aims and objectives of this dissertation. This is sometimes done using blocks shaded in grey.

Chapter 2 contains a discussion on research methodologies, more particular within an information systems context. It will motivate the choice of methodology, perspective and application used in this dissertation.

Chapter 3 comprises a literature study of transactional software development methodologies, focusing on the maintenance phase of these methodologies. Chapter 4 comprises a literature study of data warehouse development methodologies and data warehousing maintenance.

Chapter 5 contains a comparison of the challenges of maintenance between transactional and data warehousing systems as reported in literature reviewed in Chapters 3 and 4.

This comparison of the literature brings about key aspects from literature, from which an interpretive research questionnaire was developed. The development of this questionnaire and the research conducted is discussed in Chapter 6. This chapter is organised in terms of the interview questions posed, the participants interviewed and the representation and analysis of the data collected.

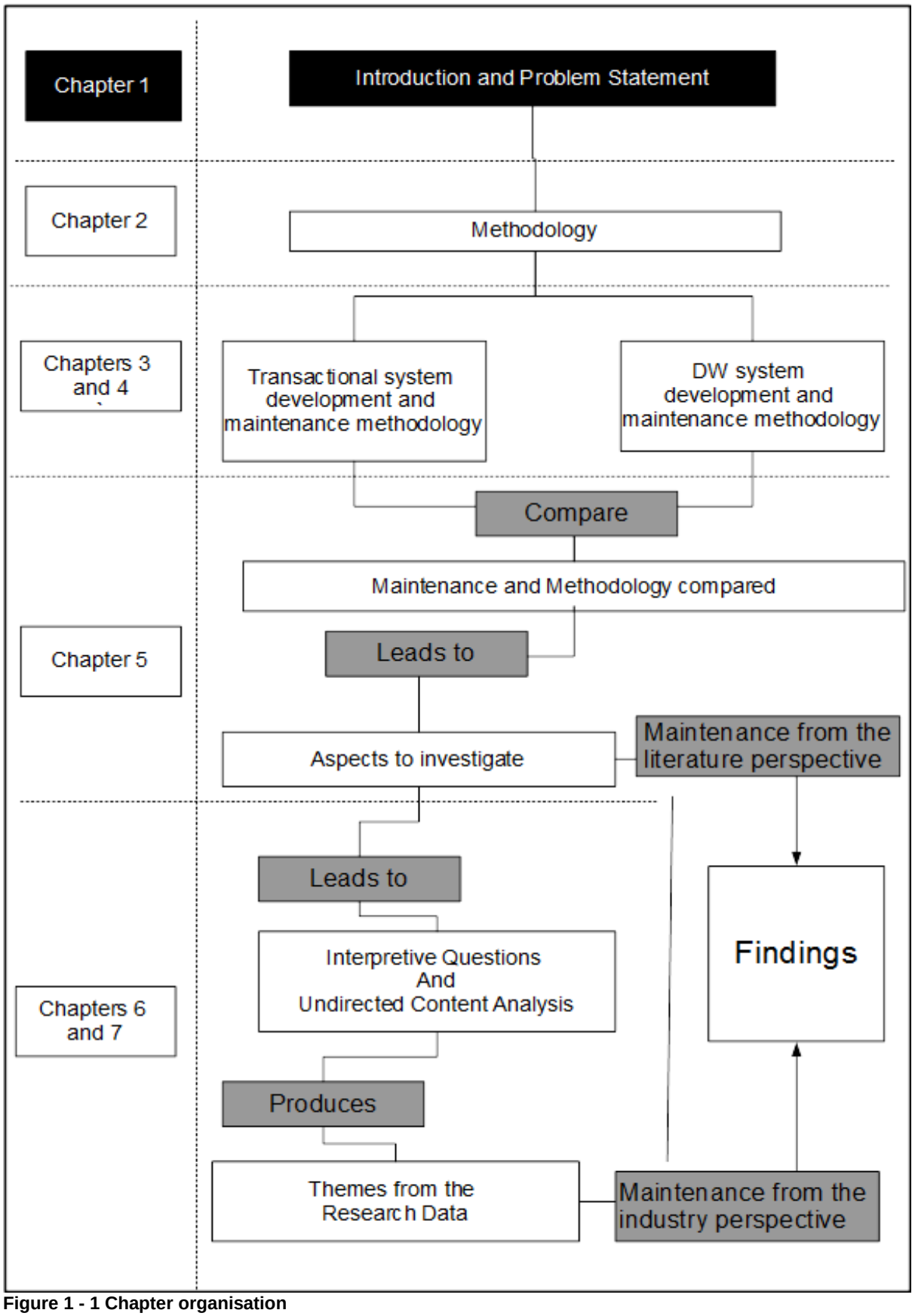
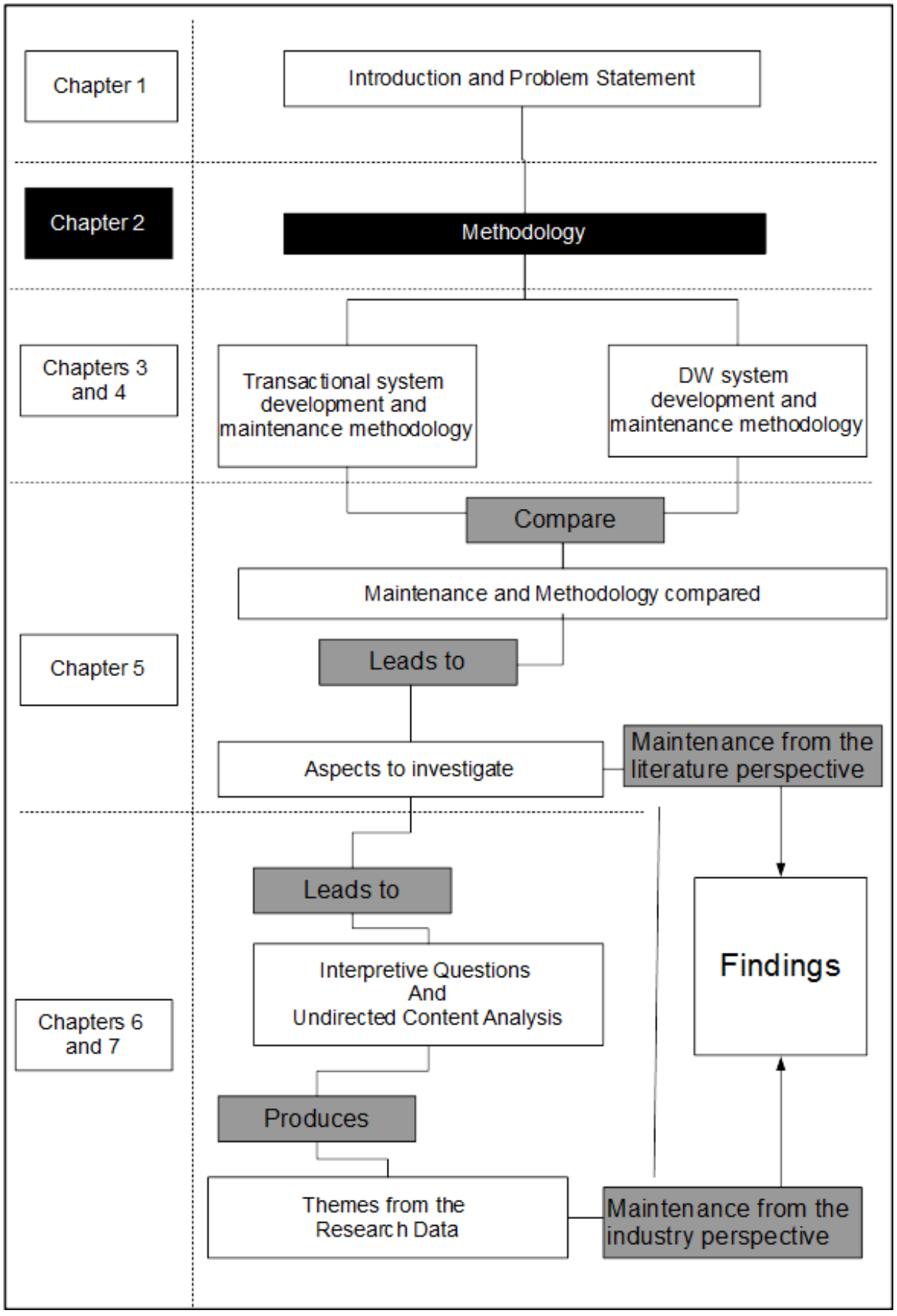


Figure 1 - 1 Chapter organisation

Chapter 7 will draw conclusions based on the comparison of data warehouse practice and software methodology practice. The chapter shall base this comparison on literature reviewed in previous chapters as well as in. It will also serves as the summary of the dissertation.

This page was intentionally left blank


Chapter 1
Introduction and Problem Statement
Chapter 2
Methodology
Chapters 3 and 4
Transactional system development and maintenance methodology
DW system development and maintenance methodology
Compare
Maintenance and Methodology compared
Chapter 5
Leads to
Aspects to investigate
Maintenance from the literature perspective
Leads to
Findings
Interpretive Questions And Undirected Content Analysis
Chapters 6 and 7
Produces
Themes from the Research Data
Maintenance from the industry perspective

Chapter 2
Research Methodology

2.1 Introduction

The purpose of this chapter is to give a description of how the research for this dissertation was performed and give justification for the choice of these methods. The reason for giving this description and justification is, as Crotty (1998: 1) describes, that there is a great amount of confusion derived from the array of methodologies and methods presented to researchers. The terminology found in different literature is often defined and described differently. The definitions and descriptions presented in this chapter are those utilised in this study.

This chapter surveys the current literature on research methodology in order to develop an understanding of the definition of research, what constitutes research and how to go about doing research. Concluding each section is a section called “This study” which specifies the choices to be made for this dissertation going forward.

This chapter begins in Section 2.2 by giving a broad definition and breakdown of research methodologies and stating where this study fits into this broad breakdown. This broad breakdown continues into Section 2.3 where research design is discussed in terms of the well-known quantitative vs. qualitative discussion.

Thereafter, research and knowledge perspectives are discussed in Section 2.4.These include the positivistic, interpretive and critical perspective on research. Based on this discussion, a choice of perspective is reasoned in Section 2.4.4.

Within each perspective, different methodologies which guide the researcher with regards to data collection and analysis exist. These methodologies are often applicable to more than one perspective; however the approach and methods used for the methodologies differ according to the research perspective. Section 2.5

discusses research methodologies and Section 2.6 discusses some of the more commonly used methods employed within these methodologies. Section 2.5.4 deduces the methodology to be used for this study whilst Section 2.6.4 discusses the method to be employed within this methodology.

This chapter is concluded with a summary in Section 2.7.

2.2 What is research?

Marczyk *et al.* (2005: 3) define two broad types of research; correlation research and experimental research. Correlation research, as the name suggests is used to determine the correlation between two or more variables e.g. the correlation (if any) between heart disease and gender. This suggests a passive type of research where data is gathered from available sources. On the other hand, it is interesting to note that the words “experiment” and “expert” derive from Latin *experiri*, meaning “try” (Free Online Dictionary, Thesaurus and Encyclopaedia). This implies that experimental research is an active form of research where something is “tried out” and data is recorded from the experiment itself.

Marczyk *et al.* (2005: 3) further report findings that the three general goals of research are description, prediction and understanding/explanation. Description refers to defining and categorizing a phenomenon. Prediction based research often tries to predict a phenomenon, which perhaps results from research undertaken to describe a noticed pattern. Understanding and explanation is research aimed at identifying the cause of a phenomenon. As such, the goal of understanding is usually accomplished by experimental research as opposed to correlation research.

2.2.1 This study

When categorising what type of study this is (correlation or experimental), it is identified that this study examines the relationship between two existing variables of a particular phenomenon, “Challenges of maintenance”. The one variable is data warehousing system maintenance and the other is transactional application system

maintenance. The data on these two variables of our phenomenon already exists – however it needs to be gathered therefore the type of research being undertaken is correlation. “Correlation” is used in a general sense and not from a statistical orientation.

The phenomenon that we are describing in this dissertation is “Challenges of maintenance and support.” We are doing a comparison of this phenomenon between data warehousing and transactional environments. Therefore our goal is the first one from the paragraph above; that is description.

2.3 Research design

Crotty (1998: 2) states that research design begins with two questions:

1. What methodologies and methods will be employed
2. How to justify this use of methodologies and methods?

It is easier to approach question 1 in individual parts: methodologies and methods. Research methods are the tools, formulae, and steps taken to gather and analyse data. Methodologies are the rules, plan and strategy of a method. More than this, methodologies deal with the desired outcomes of research and how to meet them by linking the choice of methods to the desired outcomes (Crotty, 1998: 3).

In examining the second question, Crotty (1998: 2) explains that this can only be answered by first examining the purpose of the research, and then examining the researcher’s perspective of what is considered as reality. To expand, this implies that one needs to examine the researcher’s philosophy on what the criterion are that allow a phenomenon to be considered reality rather than a mere perception. These criteria go beyond that of the researcher and into society as a whole. How do observers of the research, that being society, view the outcomes? Are the outcomes perceived or real? This boils down to epistemology – the theory of knowledge.

Therefore, to justify the methodologies being used, one needs to look at epistemological factors which inform the research perspective. The research perspective justifies the methodology chosen and finally, the methodology chosen governs choice of methods used. Therefore the two questions that Crotty (1998: 2) began with have been expanded to four questions:

"1. What epistemology – theory of knowledge embedded in the theoretical perspective – informs the research (e.g. objectivism, subjectivism, etc.)?
2. What theoretical perspective – philosophical stance – lies behind the methodology in questions (e.g. positivism and post positivism, interpretivism, critical theory, etc.)?
3. What methodology – strategy or plan of action that links methods to outcomes – governs our choice and use of methods (e.g. experimental research, survey research, ethnography, etc.)?
4. What methods – techniques and procedures – do we propose to use (e.g. questionnaire, interview, focus group, etc.)?" (Creswell, 2003: 4; Crotty, 1998: 3).

Creswell (2003: 4) used these questions to put together a practical conceptualisation of research design into three broader questions:

"1. What knowledge claims are being made by the researcher (including perspective)?
2. What strategies of inquiry will inform the procedures?
3. What methods of data collection and analysis will be used?"

Creswell (2003: 20) then puts these questions into terms of a research approach, being either qualitative or quantitative which is discussed next.

2.3.1 Qualitative, quantitative and mixed methods

Research has always been divided into two categories: qualitative and quantitative (Goubil-Gambrell, 1991: 245; Marczyk *et al., 2*005: 17; Myers, 1997: 241). There has

been varying views in recent times on where this divide exists and there also has been a school of thought combining these two research approaches into what is called “mixed method” approach of research (Creswell, 2003: 19; Johnson & Onwuegbuzie, 2004: 15; Onwuegbuzie & Leech, 2004: 772). These three categories of approach to research shall be discussed briefly here.

2.3.1.1 Qualitative methods

This involves studies that quantify their results without formal measurement (Myers, 1997: 241). Qualitative research was originally developed in the social sciences for the purpose of studying social and cultural phenomenon (Myers, 1997). Goubil-Gambrell (1991: 245) calls qualitative research “description research” as it is generally used to make observations about a specific situation. Kaplan and Maxwell (2005a: 31) state that qualitative methods are more useful than quantitative ones “when the issues being researched are not easily partitioned into discrete entities, or to examine the dynamics of a process rather than its static characteristics”.

Examples of qualitative methods are action research, case study research and ethnography. Qualitative data sources include observation and participant observation (fieldwork), interviews and open ended questionnaires, documents and texts, and the researcher's impressions and reaction (Myers, 1997: 241).

2.3.1.2 Quantitative methods

Quantitative research involves studies that make use of statistical analysis to obtain their findings. Quantitative research usually makes use of formal and systematic measurement and the use of statistics (Marczyk *et al., 2*005: 17).

Goubil-Gambrell (1991: 245) state that quantitative research usually involves experiments in which the variables are manipulated in some way and the results of the manipulation are measured and statistically analysed. As quantitative methods are experimental, the purpose of the research is usually to prove (or disprove) a

hypothesis or theory. There are usually samples chosen for experiments and these are generally chosen randomly (Goubil-Gambrell, 1991: 247).

2.3.1.3 Mixed Methods

Creswell (2003: 21) summarises the mixed methods approach to research by stating that it involves gathering both numeric (e.g. via instruments/experiments) and textual (e.g. via interviews) data. Information collected is of both a quantitative and qualitative nature.

The mixed method approaches as described by Driscoll *et al.* (2007: 20) can take either a sequential approach or a concurrent approach. The sequential approach collects quantitative and qualitative data separately e.g. doing surveys and in-depth interviews separately, then analyses separately and merges the analysis based on corresponding identifiers. The concurrent approach collects data in concurrently e.g. via a structured survey and the data was then separated into quantitative and qualitative for analysis. Results of analysis are merged to draw conclusions.

Johnson and Onwuegbuzie (2004: 19) list the advantages of methods as utilising a wider range of data collection and analysis mechanisms at the cost of requiring more time and sometimes requiring larger teams which could result in higher costs.

2.3.2 This study

After having described and defined research and research design, the outline for this research design needs to be decided upon before going into detail of the study. The study is being performed in the field of information systems. This is often viewed as a technical field, and a casual reader might automatically associate research in this field with experiments and statistics. However, Costa (2011: 2) suggests that information systems are more social than technical therefore research should follow a more qualitative than quantitative approach. This suggestion is made with the caveat that the deciding factor on research approach should be the research question(s) and the objective of the research project.

This study is of a social nature as we are studying support and maintenance challenges which deal with social interaction amongst users, the computer system and developers. Our approach to research is therefore qualitative.

2.4 Research perspectives in qualitative research

Michael Myers is the editor on the section of qualitative research in information systems for the Association for Information Systems. Myers (1997) breaks down the broad subject of qualitative research using the following diagram:

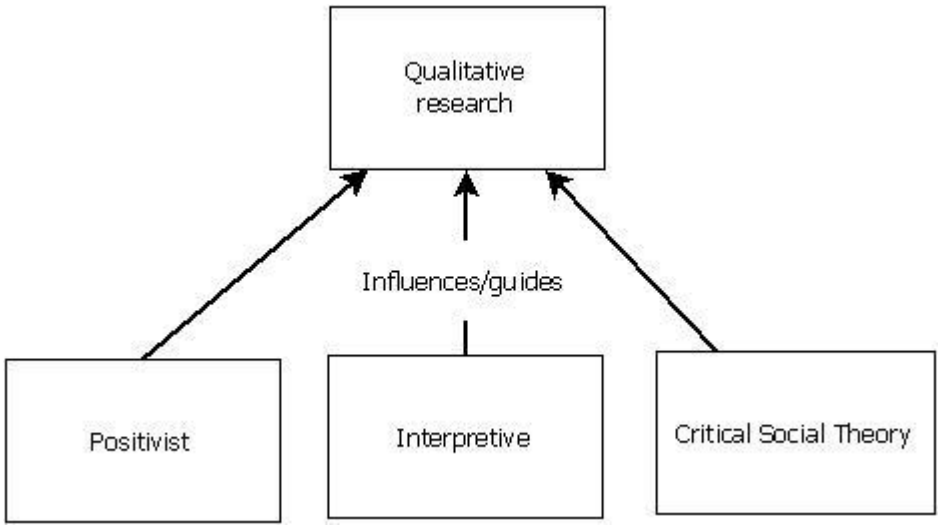


Figure 2 - 2 Perspectives of qualitative research (Myers, 1997:242)

From Figure 2-1, we can see that there are three main philosophical perspectives which influence a qualitative approach. These three perspectives, positivist, interpretive and critical social research will be discussed in more detail in the following sections.

2.4.1 Positivistic perspective

Shanks (2007: 77) describes the positivistic paradigm by stating that, from an epistemological point of view; “the researcher and the phenomena being investigated are assumed to be independent, and the researcher remains detached, neutral and objective”.

Many research scholars will disagree of placing the positivistic perspective under qualitative research. Ahrens and Chapman (2006: 822), for example, contrast qualitative research with positivistic research. However, in the same article, Ahrens and Chapman (2006: 819) state that "qualitative and positivistic researchers share many methods." Myers (1997:242) reasons that case study research and action research, which are qualitative methods can follow a positivistic perspective. Qualitative methods are therefore considered positivistic when there is a hypothesis being tested, there are quantifiable measures of variables and the positivistic paradigm is being followed.

2.4.2 Interpretive perspective

Interpretivism, in contrast to positivism, does not assume that the researcher and the phenomena being investigated are independent. The assumption is that people create and associate their own subjective meanings as they interact with the world around them (Orlikowski & Baroudi, 1991: 5). The reasoning behind this assumption is because the intention with interpretive studies is to increase understanding of phenomenon within cultural and contextual situations, where the phenomenon is examined from the perspective of the participants.

Whilst interpretivism does not suggest that research be objective, it does aim to control a bias outcome (Pather & Remenyi, 2004: 143). This is done through interpretive methodologies and data analysis techniques. One example of this is coding as utilised in grounded theory which goes back and forth between data collection and analysis until multiple behaviours indicate similar patterns and properties (De Villiers, 2005: 117).

Within an information systems context, Klein and Myers (1999: 69) state "research can be classified as interpretive if it is assumed that our knowledge of reality is gained only through social constructions such as language, consciousness, shared meanings, documents, tools and other artefacts". This definition also illustrates that

interpretive perspective allows us to perform research when data is more textual and cardinal as opposed to numeric and ordinal.

2.4.3 Critical social research

Kincheloe and McLaren (1994: 139) attribute the origins of critical research to writers Max Horkheimer, Theodor Adorno, and Herbert Marcuse from the Institute of Social Research at the University of Frankfurt (often called “the Frankfurt School”). Kincheloe and McLaren (1994: 139) report that these writers were inspired by the philosophical work by Marx, Kant, Hegel and Weber and their political views were brought about by World War I. Their major work on critical theory was published in California where they fled to due to Jewish persecution in Nazi Germany. Some time after the Second World War, Horkheimer and Adorno returned to Germany whereas Marcuse remained in the United States where he became famous amongst the student movements of the 1960s.

From the critical perspective, all knowledge, including scientific knowledge is historical and broadly political in nature (Friesen, 2008). This is because knowledge and what is seen as factual has changed due to political influences in academics. In this way, knowledge and reality is subjective. It is this subjective perspective view of knowledge which is the basis of critical research.

Unfortunately, with a subjective view, come multiple contradictory views and understandings. Often the view of a phenomenon that is accepted by society is not a democratic one and may not be one that is in the interest of society. Critical theory seeks to change this view to one shaped by social interest, especially in the interest of those who are oppressed (Friesen, 2008)

2.4.4 This study

After having discussed the epistemological perspectives relevant to the type of research, it is time to choose an epistemological standpoint before delving into the various relevant methodologies. In order to do this, the research question should again be looked at from the point of view of the three research perspectives.

The positivistic perspective begins with a theory about a phenomenon. Units of measurement are generally applied to this phenomenon to form a proposition which in turn becomes a hypothesis to test. The phenomenon in this research is "Challenges of maintenance and support." Some might consider creating units of measurement in order to measure "challenges" but in doing so, one might come across the weakness inherent in positivistic research – it does not allow for subjective realities. In order to measure how challenging a human participant finds a task, one needs to ask that participant. The answer is will always be a subjective one and one will not be able to repeat the values between multiple participants. In this sense, positivism does not work toward the goal of this research as it does not address the phenomenon of "challenges in maintenance and support". Further, we are not making a hypothesis which requires testing and experimentation. We are seeking to describe a phenomenon – hence the research goal does not, in turn, lend itself towards positivistic research.

With the chosen research phenomenon, the assumption can be made that the knowledge will only be gained via social construction. Without delving into details of methodologies and research methods, one feels that interaction with those involved in maintenance and support is required. One needs to understand their perspective of reality in order to understand what they perceive to be challenges. Therefore the philosophical perspective of interpretivism does lend itself towards our goal of describing the research phenomenon and our goal, in turn is one of an interpretive nature.

The general goal of critical social theory is to change an incorrectly accepted reality. Whilst fostering an understanding of current perception of society is part of this

critical theory, it is not the ultimate goal of the research. The ultimate goal of critical theory is to put the currently skewed perception of a phenomenon into context of its historical and political influences. The scrutiny of this skewed perception leads to alternate forms of understanding. The ultimate goal of this research for this dissertation is understanding and description. Changing of perception is not within the current scope of work for this research. Therefore, the goals of this research and critical research do not align and the critical research perspective is unsuitable.

After examining the three alternatives, interpretivism is the only perspective which aligns to the goals of this research.

2.5 Research methodologies

This section examines research methodologies, focusing on methodologies within the interpretive perspective as this perspective is being followed in this research design. Positivistic and critical social research methodology are discussed briefly to better distinguish interpretive methodology.

2.5.1 Positivistic research methodology

From a methodological point of view, Shanks (2007: 77) defines positivism as originating with theories used to generate propositions and hypotheses which are subjected to empirical testing. The researcher should be able to repeat the tests with the same results. This testing of hypotheses provides the opportunity for confirmation and falsification.

The individual terms (theory, proposition, hypothesis and hypothesis testing) in Shanks's description is further broken down. Shanks (2007: 77) builds these concepts from work by Dubin (1978), Bacharach (1989), and Lee (1989)

- A theory is made up of a set of well-defined units, laws of interaction and a boundary in which the theory holds.

- Propositions are the apparatus used to make predictions. They link the values of units in theories and are usually of "if… then…." format.
- Propositions generate hypotheses. From a single proposition, many hypotheses can be generated. This is because the terms in a single proposition are usually substituted with different values of an empirical indicator (i.e. text, pictures, animation) to form a hypothesis and this substitution can occur repeatedly.
- Hypothesis testing involves comparing predictions with observed data and looking for disconfirming evidence to falsify the hypothesis. If a hypothesis is falsified, then the hypothesis is usually refined based on the reason for falsification and subjected to further testing.

2.5.2 Interpretive research methodology

Klein and Myers (1999) have proposed a set of seven principles as guidelines for doing interpretive field research. These are summarised in Table 2.1. Whilst these principles are not to be viewed as mandatory and those involved in research should make informed decisions on which of the principles are applicable for the research in question, researchers should also guard against ignoring some of these principles as they are meant to be interdependent (Klein & Myers, 1999: 71)

1	**The fundamental principle of the hermeneutic circle** This principle suggests that all human understanding is achieved by iteration between the interdependent meaning of parts and the whole they form. This principle of human understanding is fundamental to all the other principles.
2	**The principle of contextualisation** Requires critical reflection on the social and historical background of the research setting, so that the intended audience can see how the current situation under investigation emerged.
3	**The principle of interaction between the researchers and the subjects** Requires a critical reflection on how the research materials (or "data") were socially constructed through the interaction between the researchers and the participants.
4	**The principle of abstraction and generalisation**

	Requires relating the idiographic detail revealed by the data interpretation through the application of principles one and two to the theoretical general concepts that describe the nature of human understanding and social action.
5	**The principle of dialogical reasoning** Requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings ("the story which the data tells") with subsequent cycles of revision.
6	**The principle of multiple interpretations** Requires sensitivity to possible differences in interpretations among the participants as are typically expressed in multiple narratives or stories of the sequence of events understudy. They are similar to multiple witness account, even if all tell it as they saw it.
7	**The principle of suspicion** Requires sensitivity to possible "biases" and systematic "distortions" in the narratives collected from the participants.

Table 2 - 1 Summary of principles for interpretive field research (Klein & Myers, 1999: 72)

The seven principles summarised in Table 2-1 are expanded below based on the discussions of these principles by Klein and Myers (1999: 71)

1. <u>The fundamental principle of the hermeneutic circle</u>

This principle is the foundation upon which the other six expand. It suggests that any understanding we have of a phenomenon is broken down into many smaller parts to which meaning is attached. The meaning of each individual part is then interrelated to give the meaning of the entire phenomenon. The meaning attached to the individual parts may have many different variants. An example given by Klein and Myers (1999: 71) is a phenomenon being researched with multiple participants. The individual parts are the different initial understandings of the participants. Only once we have iterated through each variant of the meaning of the individual parts and the interrelationship of these variants, can we gain a holistic understanding of a phenomenon. In Klein and Myers (1999: 71) example, this means putting together multiple participants' understanding to gain a "complex whole of shared meanings".

2. The principle of contextualisation

Klein & Myers (1999: 73) explain that the basis of this principle is that an interpretation of a phenomenon is influenced by its historical context. As opposed to positivistic research where the philosophy regarding historical context is that results should be repeatable, the interpretive philosophy is that results of research differ as time goes by and society changes (Klein & Myers, 1999: 74). Another aspect to this principle is that the research subjects and participants shape history in the sense that actions today may change the viewpoints of the future. In this way, the researcher needs to set the subject matter in its social and historical context so that the intended audience understands how the current situation under investigation has emerged.

3. The principle of interaction between the researchers and the subjects

The researcher needs to be aware that the interaction with participants in research affects the researcher's perception of the subject matter and therefore could affect the results of research. This change in perception may or may not be part of the planned research process, but one needs to be aware of the affect that the research is having on one's perception. As Klein and Myers (1999: 74) explain, the reason for this is that, with interpretive research, facts are produced as a result of social interaction with participants. Participants are also interpreters of data as the interaction with the researcher can their perception of the subject. Participants are also analysts because the research can change their action as a result of the change of their perceptions.

4. The principle of abstraction and generalization

The idea behind this principle is that a unique phenomenon within a particular context may be abstracted from that context and compared to generalised abstract theories that apply to multiple situations (Klein & Myers, 1999: 75). It should be noted that the purpose here is not to add validity to a theory in a statistical or positivistic sense by comparing it to other similar result sets. The purpose is rather to illustrate logical reasoning used by generalising to "social theories such as structuration theory or actor network theory" (Klein & Myers, 1999: 75).

5. The principle of dialogical reasoning

In interpretive research, one does not require objectivity. A bias view on a phenomenon is a starting point of understanding. However, a researcher should not allow that initial bias view skew the results of research and analysis of data during the research. To prevent this, Klein & Myers (1999: 76) emphasize that the researcher needs to make philosophical assumptions transparent to the audience as well as to himself or herself, as well as the interpretive methods chosen and their strengths and weaknesses.

6. The principle of multiple interpretations

When doing research and gathering data from many participants, it is common to come across differing and even conflicting viewpoints of the phenomenon being researched (Klein & Myers, 1999: 77). It is up to the researcher to analyse these differing viewpoints in the information gathered and alter the results of the research in a manner which takes these conflicting viewpoints into account.

7. The principle of suspicion

Klein and Myers (1999: 77) draw on the work of Ricoeur, Marx and Freud by explaining that the idea around this principle is to "reveal the effects of socially created distortions (Marx) and psychopathological delusions (Freud)." This principle is similar to the previous six in that it encourages critical thinking but is distinct from the previous six principles as it concerns the researcher with the discovery of "false preconceptions" of participants (Klein & Myers, 1999: 77).

2.5.3 Critical social research methodology

Critical social research requires self-reflection on pre-suppositions being made, both personally by the researcher and holistically by society (Kincheloe & McLaren, 1994: 140). Thus, the researcher's assumptions form part of the investigation and are subject to change due to further analysis and understanding.

Fossey *et al.* (2002: 720) explain that understanding is a goal of critical research, but as a stepping stone to be used in the transformation of society. As such, methodologies used by the critical perspective aim to foster self-reflection, mutual learning, participation and empowerment, rather than the acceptance of discoveries.

Friesman (2008) provides a guideline to critical theory research with the following outline to a methodology:

- Identifying ideas or claims that are presented as obvious, inevitable, or matter-of-fact in dominant bodies or sources of knowledge.
- Scrutinizing these ideas or claims in the context provided in other more marginal knowledge forms or sources.
- Revealing through this scrutiny that behind dominant claims and ideas lay one or more politically-charged and often contradictory ways of understanding the issue or phenomenon in question.
- Using this underlying conflict as the basis for developing alternative forms of understanding and point to concrete possibilities for action.

2.5.4 This study

The interpretive perspective has already been earmarked as the philosophical perspective taken for this study. Klein and Myers's seven principles guide the researcher and provide a checklist to aid the researcher in ensuring correctness and comprehensiveness. These seven principles do not pose any contradictions to the topic and study being undertaken.

When examining positivistic methods, we again see that the methodologies are based on an initial hypothesis and measurable data. When examining critical methodologies, we again see that the methodology is geared toward challenging norms and traditions. These pre-requisites are not fulfilled by the study being undertaken which deals with investigation, description and comparison between two phenomena.

The study therefore follows the interpretive perspective and the methodology which guides the underlying methods will also be interpretive. The principles proposed by Klein and Myers (1999) shall be used when conducting the research and drawing conclusions in Chapter 6 and Chapter 7. A reflection of the study is given in Chapter 7 based on these principles.

2.6 Research methods

Research methods are often more suited to particular research methodologies. Hence the discussion that follows have divided the research methods into the types of methodologies already mentioned; positivistic, interpretive and critical research.

2.6.1 Positivistic research methods

The positivistic methods discussed are the scientific method in Section 2.6.1.1 and research surveys in Section 2.6.1.2. Both of these can be accomplished with minimal interaction with the researcher enabling the researcher to remain impartial to the research participants' perspective. Scientific method involves proposing a hypothesis and proving it or disproving it. Surveys involve studying responses from research participants.

2.6.1.1 Scientific method

Burns (2000: 3) states that this method "has been the conventional approach to research in all areas of investigation". Walizer and Wienir, (1978: 231) refer to the classical experiment as the "grandmaster of research designs because of the relatively simple way the researcher can deal with many of the problems of demonstrating causation." As such, this method from the natural sciences which utilizes measurement and control has found a place in the social sciences and the study of human interaction and society.

Burns (2000: 105) characterizes the scientific method of research by being designed around a hypothesis. The following are the three essential steps of the scientific method:

" 1. The proposal of a hypothesis to account for a phenomenon.

2. The deduction from the hypothesis that certain phenomena should be observed in given circumstances.

3. The checking of this deduction by observation." (Burns, 2000: 106)

Walizer and Wienir (1978: 232) outline the main objective of the experiment as the observation of a relationship between two variables over a period of time. This is consistent with Burns's steps of the scientific method as observation is the final step of the method.

The following are relevant when designing an experimental procedure:

- Independent variable/treatment: This is the variable that is the first to react in a comparison between two variables (Walizer and Wienir 1978: 232). Burns (2000: 124) describes the independent variable as one that reacts first to manipulation in the experiment. When the variable is manipulated, it is referred to as the treatment or stimulus (Walizer & Wienir, 1978: 232; Creswell, 2003: 165).
- Dependent variable: Walizer and Wienir (1978: 232) define the dependent variable as the variable that reacts depending on the way in which the independent variable acts. Burns (2000: 125) explains that this reaction is what is measured.
- Control group: This is the group that contains very little or none of the treatment variable (Walizer & Wienir, 1978: 232).
- Experimental group: This group is manipulated to contain a high degree of the treatment variable (Walizer & Wienir, 1978: 232).
- Null hypothesis: This is a hypothesis generally proposed for the purpose of being proved incorrect. Burns (2000: 112) explains that a null hypothesis takes the format of stating that there is no consequential difference or relationship between phenomenon being studied and any difference noted is coincidental.

- Experimental hypothesis: “The two conditions will not differ by an amount which is greater than is to be expected by chance alone” (Burns, 2000: 113).
- Pre-test: The measurement of depended variable prior to the independent variable being manipulated (Walizer & Wienir, 1978: 232).
- Post test: The measurement of dependent variable after manipulation of the independent variable (Walizer & Wienir, 1978: 232).

Burns’s (2000: 105) three essential steps of scientific method above are expanded to use the null hypotheses.

1. Create the null hypothesis by stating, as Burns (2000: 113) describes that any relationship between dependent and independent variables are coincidental.
2. After manipulation of the independent variable, and observation of dependent variable, calculate the probability of the change that has occurred in dependent variable occurring. Burns (2000: 113) notes that this calculation can be achieved using a theoretical sampling distribution.
3. Burns (2000: 113) explains that we may accept the null hypothesis if the probability calculated in Step 2 is sufficiently high that we may attribute the change in the dependent variable to coincidence and not to manipulation of the independent variable.
4. “Accept the experimental hypothesis if the null hypothesis has been rejected. Reject the experimental hypothesis if the null hypothesis cannot be rejected” (Burns, 2000: 113).

2.6.1.2 Surveys

Creswell (2003: 153) defines a survey as studying a sample of a population in order to generalise and make claims about the entire population based on a numeric description of trends, attitudes and opinions of the sample.

Creswell (2003: 154) divides the survey research method into the following parts: the survey design, the population and sample, instrumentation, variables in the study and data analysis. These parts are summarised below.

The survey design:

The researcher needs to indicate the purpose of research and why a survey is the preferred type of data collection process. As part of the survey design, the time period for data collection should be specified as well as the data collection method. Time period may be one point in time (cross-sectional) or over a period of time (longitudinal). Examples of data collection methods include questionnaires, structured interviews and web-based surveys.

The population and sample:

The size of the population and means of identifying individuals in the population should be described as well as the sampling strategy for the population. Sampling can either be single or multi-stage. For positivistic research, sampling should be random where each individual in the population has an equal chance of being chosen unless stratification takes place. Stratification is a process that ensures specific characteristics of the population represent a specified ratio in the sample. Stratification strategies should be elucidated.

Instrumentation:

The actual survey instrument to be used should be discussed by the researcher. This includes the design of the instrument, the history of the instrument in previous studies and its applicability to the research in question.

Variables in the study:

It is recommended to relate the variables of a study to specific questions. For example if doing a survey study on challenges of maintenance in an IT department, a variable could be occupation title/level of participant (i.e. manager, developer, helpdesk). The questions asked in the survey could vary based on this variable. These questions in turn could pertain to different hypotheses being tested. It is thus recommended that a table be drawn up showing the variable, hypotheses and question on survey.

Data Analysis:

Data analysis begins by analysing the responses and non-responses. This involves determining if there is a skew in the respondents and how that skew might bias the result. Analysis should include discussion on variables, means, averages and standard deviations of these variables. The computer program and/or formulae being used for statistical analysis should be described as well as reasons behind use of these tools.

2.6.2 Interpretive research methods

Interpretive methods allow the researcher to be more involved with the research participants in order to gain an understanding of the participants' understanding of the research topic. Ethnography, discussed in Section 2.6.2.1 requires the researcher to immerse himself into the research participants' environment. Grounded theory, discussed in Section 2.6.2.2 involves theory development through continuously interpreting participants' perspectives. Content analysis, discussed in Section 6.2.2.3 involves systematically analysing transcriptions of interactions with participants. Content analysis is versatile can be used formulate a research questions as well as to conduct research where the research goals already exist.

2.6.2.1 Ethnography

Myers's (1999) paper serves as a tutorial on the use of ethnography in information systems. The following summary is based on Myers's work.

Ethnographers attempt to become part of the research phenomena's social context and data is collected through participant observation. Myers (1999: 5) discusses the benefits and limitations of ethnographic research. The benefit of ethnography is the in-depth understanding that the researcher gains. The main disadvantage or limitation of ethnography is the extended period of time required in the field (Myers, 1999: 6; Iqbal *et al.* 2005: 34) to collect data as well as the extended time required

to analyse the data. Hughes *et al.* (1994: 434) note difficulties and limitations in scaling ethnography out of small contexts.

Atkinson and Hammersley (2007: 150) assert that when doing ethnographic research, one needs to be disciplined in writing up notes and summarising interviews as and when they happen. A system should be developed and maintained from the start of research that indexes and categorises data collected due the volume of data that ethnographers deal with. This is because ethnographic research can take long periods of time, collecting vast amounts of research data whereby important details may be forgotten (Hammersley & Atkinson., 2007: 151). It is also important for ethnographers to regularly review their notes and work as it is not uncommon for a researcher's views and understandings to change during the research process. The other reason for regular review is that there is too much to review and correct at the end of the research process and this could significantly delay a write up (Myers, 1999: 10).

An ethnographic research write up often takes the form of a book or doctoral thesis due to the amount of data collected. This poses a problem for ethnographers wanting to publish journal articles which are much shorter in length (Myers, 1999: 10). Journal articles usually discuss a particular theme or phenomenon whereas the ethnographic researcher may have many different phenomena and themes to write about. Myers (1999: 10) advises that ethnographers therefore need a write up strategy which splits the research write up in such a way that individual parts of it may be published separately. In this way, there is potential to publish many articles from one piece of field work.

When evaluating ethnographic research, the following points are identified by Myers (1999: 11):

- The findings of the research is new and a contribution to the field.
- The research should offer rich insights to the subject matter.

- It should be evident that a significant amount of data has been collected and the subject matter should be set in its social and historical context with multiple viewpoints expressed.
- The research methods used should demonstrate how and why the research should be considered valid by reviewers.

2.6.2.2 Grounded Theory

"A Grounded Theory is a set of integrated conceptual hypotheses systematically generated to produce an inductive theory about a substantive area" (Glaser & Holton, 2004: 3). Glaser and Strauss (1967: 1) defined grounded theory as "the discovery of theory from data"

Barney Glaser is the co-inventor of grounded theory (Adolph, Hall & Kruchten, 2011). His work with Judith Holten serves as a preliminary guide to grounded theory (GT) (Glaser & Holten, 2004) where they define the goal of GT as a conceptual theory, abstract of time, place and people.

The following is a summary the elements of GT using this work by Glaser and Holton (2004) as a template.

1. Theoretical Sensitivity

Glaser and Holten (2004: 11) list theoretical sensitivity as the first element of the GT methodology. The researcher doing grounded theory must have the temperament to remain analytical and tolerant of confusion during the research time period. The researcher must also be able to use this tolerance and temperament to conceptualise abstract connections and variances into a theory (Glaser & Holton, 2004: 11; Glaser &Strauss, 1967: 46). Glaser and Strauss (1967: 46) further suggest that "theoretical sensitivity is in continual development". The researcher should not have predetermined ideas – the theory should evolve from the data.

2. Getting Started

Glaser and Holten (2004: 11) and (1990: 5) emphasise that data collection and coding should start immediately, even prior to forming a research question or doing a literature review. The reason for this is that the research question should come from listening to participants, and not from the researcher's pre-conceived hypothesis of a problem.

3. All is data

There does not need to be a defined method for collecting data (Glaser & Holton, 2004: 12). Whilst the most popular data collection choice tends to be interviews, the researcher may choose to include observations and other non-elucidated techniques (Glaser & Holton, 2004: 12) as these techniques become clearer as more data is collected and analysed (Glaser & Strauss, 1967: 45), (Corbin & A Strauss, 1990: 6).

4. Use of the literature

Glaser and Holton (2004: 12) suggest delaying a literature review as not to be influenced by theoretical ideals. A researcher might also fail to recognize a new phenomenon by relating it to phenomena described in the literature. Once the basic premise of GT has been fulfilled, that is a theory has been generated from the data, and its development is well underway, then a literature review may ensue. The literature should be treated as just another source of data to be integrated into comparative analysis.

5. Theoretical coding

"The conceptualization of data through coding is the foundation of GT development" (Glaser & Holton, 2004: 12). Corbin and Strauss (1990: 4) explain that groups and codes are provisionally created by noticing repetitions in the data. This data is broken down and grouped into codes which evolve into the theory explaining the data. Hypotheses are generated from conceptualising how codes relate to each other (Glaser & Holton, 2004: 12).

6. Open coding

Open coding, described by Glaser and Holton (2004: 13) tests the researcher as the process involves going through the data line-by-line seeking out codes by interrogating the data. Data interrogation takes the form of questioning “what is this data a study of?”, seeking out repeating categories and themes in participants concerns and seeking out from the data how these concerns are being addressed (Glaser & Holton, 2004: 13; LaRossa, 2005: 841). The patterns found yield codes. As constant, on-going coding continues, ideas are stimulated and theories emerge from the data – and importantly, not from the researcher’s preconceptions.

7. Axial coding

Axial coding is summarised by La Rossa (2005: 846) as a deeper analysis of an existing category. Attention is specially given to inter-relationships between sub-categories and the parent-child relationship. “The focal category or variable is temporarily placed at the hub of the analysis and the when, where, why, and so constitute the spokes around the hub” (LaRossa, 2005: 847).

8. Theoretical sampling

In theoretical sampling, the researcher continues to search for data based on emerging theories and hypotheses. The main criteria for data collection are categories and properties that are incomplete and thereby warrant further data collection. Based on this incomplete data, Glaser and Holton (2004: 12) explain that decisions are made on where to go searching for further data and how theoretical sampling allows the researcher to control the data collection according to relevance.

9. Constant comparative method

“All that GT is, is the generation of emergent conceptualizations into integrated patterns, which are denoted by categories and their properties. This is accomplished by the many rigorous steps of GT woven together by the constant comparison process, which is designed to generate concepts from all data” (Glaser, 2008).

Glaser’s statement above summarises the core goal of the constant comparison method. During coding, incidents (or phenomena), concepts (or themes) and hypotheses emerge. Glaser and Holton (2004: 12) impart that the constant comparison method involves comparing incidents to each other to establish where

and how these incidents occur. This comparison develops into concepts and hypotheses. The concepts are then compared to more incidences to develop understanding of properties which lead to further hypotheses. These concepts are verified and then compared to each other. This establishes a best fit between many choices and integrates similar hypotheses – which develops into theory.

10. Theoretical saturation

With continual application of the constant comparison method, LaRossa (2005: 841) demonstrates how new indicators which define concepts continue to emerge. These indicators provide insight into the construction of a concept. Theoretical saturation occurs when addition of an indicator provides no new insight into the concepts construction. “A theoretically saturated concept is a well grounded concept” (LaRossa, 2005: 841).

11. Core variable

During comparisons, a core theme emerges which accounts for most of the variations other themes. This is the “core variable” (Glaser & Holten, 2004: 14), or “core category” as referred to by (Corbin & Strauss,(1990: 17) both refer to the variable/category around which the study will focus. More coding will be needed to verify the core variable. The verification involved is identifying this variable in large proportions of the data and relating it to other themes in the data.

12. Selective coding

Glaser and Holten (2004: 15) as well as Corbin and Strauss (1990: 18) report that once the core variable is identified and verified, coding should be limited to only those variables that significantly relate to the core variable.

13. Delimiting

Delimiting refers to limiting, or restricting the theory that is evolving from the data. Glaser and Holten (2004:15) explain that delimiting “occurs at two levels – the theory and the categories.” Delimiting occurs on the theory level as fewer changes are made to the evolving theory and finer changes are made to the properties surrounding the theory. This delimiting leads to reduction which is when the

researcher discovers uniformity among the properties of the theory and the theory can be reformulated with a smaller number of higher level properties.

Glaser and Holten (2004:15) continue by explaining that delimiting occurs on the category level by reducing the number of categories for coding. This occurs naturally as the broadness of the theory narrows down and the researcher focuses on the core category.

14. Inter-changeability of indicators

Incidents or phenomena are indicators of research concepts which apply to themes or research categories. During data collection and analysis, these indicators are interchanged within and applied to different themes and compared. Glaser and Holten (2004: 16) rationalise that this interchanging continues until codes become verifiable and saturation is reached.

15. Pacing

Glaser and Holten (2004: 16) relate that grounded theory generation requires patience and persistence. Data collection, followed by line by line coding and iterative selective coding is tedious and could easily tempt the researcher to apply a bias to a pre-conceived theory. However, forcing a theory could leave the research unverifiable. It is important to continue with the process until theories present themselves.

16. Memoing

Memos should be captured whenever ideas occur whilst going through the data as these ideas shall connect phenomena with the themes which lead to hypotheses (Glaser & Holten, 2004: 17). Constant writing of memos slow down the research pace, however this is necessary for the researcher to rethink ideas. Omitting memos shall result in a “less elaborated and satisfying integration of the analysis” (Corbin & Strauss, 1990: 10).

17. Sorting and writing up

After categories and themes have been saturated, memos relating to the core category should be identified. Glaser and Holten (2004: 18) place emphasis on sorting out these memos as this sorting puts together the framework for the write up and ensures that there are no pre-conceived ideas sneaking in.

18. Analytic rules developed during sorting

The emphasis that Glaser and Holten (2004: 18) place on sorting the memos is further expanded by setting out "Analytic rules developed during sorting". Rules of sorting are essential to keep the process long ordered and disciplined so as to ensure themes are developed properly into theory. The following is a summary of these rules set out by Glaser and Holten (2004: 18):

- The starting point for sorting of memos will present itself in the core variable as other ideas will relate to this. Sorting should not be from beginning to end.
- Whilst sorting, new ideas will present themselves – these should be put into memos and integrated into the sort.
- Each concept that is introduced to the theory is carried forward to subsequent sorts. If an idea is not properly introduced to the developing theory by, it should not be included in any of the sorts.
- Ideas and memos that are not linked to the core category should be discarded or the researcher runs risk of breaking out of the developing theory. Alternatively, the core category needs to be altered to include the idea contained in the memos.

Concluding remarks

Burck (2005: 246) summarises grounded theory well by stating "grounded theory lends itself to the exploration of under-theorized areas" and "The strength of a grounded theory approach lies in its ability to aid a researcher to generate theory about processes and to develop conceptual analyses of social worlds".

2.6.2.3 Content Analysis

Zhang and Wildemuth (2009: 1) report the following definitions of qualitative content analysis:

- “a research method for the subjective interpretation of the content of text data through the systematic classification process of coding and identifying themes or patterns” (Hsieh & Shannon, 2005: 1278)
- “an approach of empirical, methodological controlled analysis of texts within their context of communication, following content analytic rules and step by step models, without rash quantification” (Mayring, 2000: 2)

Zhang and Wildemuth (2009: 3) have put together a set of steps to follow when doing content analysis. These are flexible but take one through the general process, from preparing the data to writing up the findings. The following is a summary of these steps.

Step 1: Prepare the data

Data collected (e.g. interviews) should be transformed into written text. Zhang and Wildemuth (2009: 3) describe how the level of detail of data transformed should be based on how much of the data relates to the research question. Another consideration is whether external factors, e.g. mood of the participant should be transcribed.

Step 2: Define the unit of analysis

This is the basic unit of text to be classified. Coding is based on the basic unit and this step is fundamental to the content analysis process (Graneheim & Lundman, 2004: 106; Zhang & Wildemuth, 2009: 3). Based on how the data relates to research question, the basic unit may be individual words, phrases, paragraphs, themes or even an entire document or interview. Schilling (2006: 31; 2009: 31) as well as Zhang and Wildemuth (2009: 3) suggest that when using themes as the basic unit, a code might be assigned to various amounts of text as you are looking for an expression of an idea.

Step 3: Develop categories and a coding scheme

The three sources for a coding scheme are: the data, previous related studies and theories (Zhang & Wildemuth, 2009: 3).

Where no theories are available, categories are generated from the data inductively. Inductive content analysis is often used for theory generation. This is also known as conventional content analysis (Hsieh & Shannon, 2005: 1279). Here, Zhang and Wildemuth (2009: 4) recommend Glaser and Strauss's (1967) constant comparative method as it encourages original insights and highlights the differences between categories. The essence of this method is that texts within categories are constantly compared to understand the categories better. These categories are constantly integrated as their similarities become evident.

Zhang and Wildemuth (2009:1) and Mayring (2000: 4) define deductive content analysis as using content analysis on research that is being based on an existing theory. Hsieh and Shannon (2005: 1281) refer to this as directed content analysis. In these cases, the theory's coding categories may exist and can be used as a base on which to add further categories as they emanate from new data. The advantage of using already existing coding categories is that the constant comparison has already been performed and categories being dealt with are closer to core categories.

Zhang and Wildemuth (2009:4) emphasise the importance of developing a coding manual consisting of category names, definitions or rules for assigning codes, and examples. With the constant comparative method, the coding manual should evolve throughout the process, with interpretive memos, explanations and notes.

Step 4: Test Your Coding Scheme on a Sample of Text

To avoid detrimental errors later on in the process, the coding should be tested and verified early in the process using a sample of the data. When working in a research team, Zhang and Wildemuth (2009: 4) advise that testing should be verified by different team members and any differing points of view of team members should be discussed and resolved with the research team. Schilling (2006: 33) proposes that

testing take the form of independent testers coding only with coding instructions. Feedback from these independent testers should be utilised to improve the coding scheme. Testing should be repeated until sufficient coding consistency is achieved (Zhang & Wildemuth, 2009: 4).

Step 5: Code All the Text

After the coding scheme has been tested and verified, Zhang and Wildemuth (2009: 5) direct that one should proceed to code all the text, whilst constantly referring to the coding manual so that all text is coded uniformly. It is not unlikely that new codes will need to be created as new data is collected.

Step 6: Assess the coding consistency

Zhang and Wildemuth (2009:5) warn that errors and inconsistencies may occur due to fatigue, new codes or even a subtle change in the researcher's perspective. Thus, there is a need to periodically assess coding consistency. Schilling (2006:33) prescribes checking the codes between 10 and 50 percent through the coding process and then again about two thirds the way through.

Step 7: Draw Conclusions from the Coded Data

Conclusions are drawn by reconstructing meanings derived from data. This is done by comparing and exploring themes of categories and their properties (Schilling, 2006: 34; Zhang & Wildemuth, 2009: 5). Success in this step relies on the researcher's reasoning abilities.

Step 8: Report your methods and findings

Zhang and Wildemuth (2009 :5) explain that to validate the research, the methods and coding practice must be included in a write up. As the results being reported are not of a statistical nature, reporting findings is challenging. Presenting the findings, which may include patterns, themes and categories, is usually done by use of quotations. However, depending on the research done, graphs, charts and

conceptual networks may also be used. The researcher should aim to strike a balance between describing the data and giving interpretations of it.

Concluding remarks

Content analysis is a method flexible enough to be used from a qualitative and quantitative approach (Graneheim & Lundman, 2004: 1; Mayring, 2000: 3). Hsieh and Shannon (2005: 1286) summarise their research by tabulating how types of content analysis can, by interpreting text be used to generate theory, work on existing theories as well as be used for descriptive purposes.

2.6.3 Critical social research methods

As critical social research involves inducing change in an environment, the methods used require taking action. The aptly named research method, action research seeks to achieve this. Section 2.6.3.1 discusses action research more specifically aimed at crafting change in an information system environment.

2.6.3.1 Action research

Action research is essentially a two stage process with the first stage diagnosing the social situation by collaborating with the subjects of research. The second stage involves agreeable change to the social situation and studying the effects of this change (Baskerville & Myers, 2004: 329).

Baskerville and Myers (2004: 331) report four key premises of action research:

1. Consequences define human action - (Charles S. Peirce (1839-1914)). Humans base their actions on consequences of this action and thus actions have a purpose. If people are prevented from choosing their own actions, then people lack a purpose.
2. Truth is embodied in practical outcome - (William James (1842 – 1909)). Human thought is revealed in human action. As humans have different perceptions of reality, a society's reality may be revealed by their practices and actions.

3. Logic of Controlled Inquiry - (John Dewey (1859-1952)). In society and with individuals, the approach taken to change existing conditions occurs when ideas take on logical forms in a process called controlled inquiry. Once a problem occurs, ideas develop which offer possible solutions. Thereafter, experimental actions bring about the desired change.
4. Human action is contextualized socially – (George Herbert Mead (1862-1931)). Humans change their actions based on how society reacts to these actions and thus actions are socially relative. Therefore, the action researcher performing actions should be observed by collaborative team members who are situated in the social context.

Baskerville (1999: 13) reports on Susman's and Evered's (1978: 588) five phase cyclical process as the most prevalent one for action research. After development of a client-system or research environment, the following five phases are iterated:

1. Diagnosing
2. Action planning,
3. Action taking,
4. Evaluating and
5. Specifying learning.

Baskerville's work is aimed at those doing research in information systems and hence there is mention of a client. When research is done pro bono or for academic purposes, the "client" as referred to by Baskerville would be the community in which the research takes place. The following is a summary of this process guided by Baskerville (1999: 14), beginning with the client-system environment followed by the five iterative steps.

Client-System Infrastructure

This refers to the agreement that sanctions the actions that researchers may specify. The boundaries of the research domain are also defined.

Diagnosing

Diagnosing involves identifying the primary problems and causes of these. A hypothesis is formed of the nature of the organisation and the nature of the problem.

Action Planning

Participants and the researchers collaborate to specify actions aimed at relieving these organisation-wide problems. The plan should include the desired future state and what changes would achieve this state.

Action Taking

Again, collaboration is needed to implement the plan.

Evaluating

After completing actions, those involved collaborate to evaluate whether the outcomes relieved the problems. If the problems were solved, it should be assessed whether the implemented actions were the only factor involved in the solution. Problems that weren't solved should go back to the diagnostic phase.

Specifying Learning

Knowledge gained during the action research iteration should be shared and reused. On successful iterations, it is important for the society in the research context to benefit from the knowledge gained by reflecting the changes implemented. On unsuccessful iterations, lessons learnt may serve as a foundation for a change in hypothesis for the next iteration of the research. Whether the iteration was a failure or success, the scientific community should benefit from a write up of the research.

2.6.4 This research

An epistemological perspective i.e. interpretivism, has been chosen in Section 2.4.4. The method of research will therefore follow an interpretive methodology based on this perspective.

Ethnography is a long term commitment in which a theory is generated from observation. The researcher needs to spend a long period of time in one social context, becoming one with the society or community. This research involves a very

broad and scattered community of software developers and data warehouse developers. The different companies from which individual participants work would form multiple different social contexts and it would not be feasible to immerse oneself within these companies for extended periods of time. Therefore, it is not feasible to consider doing ethnographic research.

When doing grounded theory research, the research phenomenon is something that develops along with a theory as observations occur. In this research, the phenomenon is already chosen; that being “the challenges of maintenance”. Further, when doing grounded theory research, data collection and coding should start immediately – without influence from literature. In this research, we need to first do a literature review to understand the topics of software development, software maintenance, data warehouse development and data warehouse maintenance. Thereafter, the data collection and analysis ensues and conclusions will be drawn based on correlating data collected with available literature.

Content analysis is a flexible methodology for the interpretive analysis of textual data. Data is transcribed based on relevancy. The definition of the unit of analysis depends on how the data relates to the research question. This gives the researcher flexibility in coding and allows for consideration of external influential factors. Unlike grounded theory, content analysis encourages the use of previous studies, where a researcher may re-use coding categories which have been used in similar research in order to build on those theories. Content analysis is therefore an applicable methodology to this study.

2.7 Data collection

The various methods discussed in Section 2.6 may have multiple associated data collection processes. Ethnographers generally employ both observation and interviews (Atkinson & Hammersley, 2007: 110). Surveys may collect data through self-administered questionnaires, interviews or the survey may be less personal e.g. an anonymous internet survey (Creswell, 2003: 156). Burns (2000: 118) defines measurement as “the operation of assigning numbers to observed events (usually a

subject's responses)." Using this definition of measurement, we can describe data collection for the scientific method as regular measurement of the relationship between research phenomena. This measurement usually takes on a statistical function, e.g. ratio, mean, summation.

We shall discuss two more common qualitative data collection techniques: Interviews and observation.

2.7.1 Interviews

There are three types of interviews: structured, unstructured and semi-structured interviews.

Structured interviews are put together so that there is uniformity of questions, time allocated and interviewer reaction between each interviewee. Fontana and Frey (1994: 364) explain that the idea is to remain neutral to the interviewee's responses in order to allow the interviewee to remain uninfluenced and provide rational responses.

Unstructured interviews are by definition, less prescribing. Depending on the method and methodologies being undertaken, the interviewer may wish to blend and imitate the attire of the interviewee as well as hold the interview in a surrounding familiar to the interviewee or the interviewer may decide to present himself (or herself) as the researcher and remain detached (Burns, 2000: 411; Fontana & Frey, 1994: 366). In either situation, the interviewer needs to be able to establish a rapport with the interviewee as the questions are generally of the open-ended type. In order to collect sufficient data from these types of interviews, the interviewee will have to feel comfortable enough to speak of the subject being discussed without too much influence from the interviewer.

Fontana and Frey (1994: 365) state that both structured and unstructured interviews may be used in group interviews. Focus groups, often used in product manufacturing involve groups of people being interviewed for their opinions towards a product or phenomenon. Focus groups are an example of structured group interviews.

Unstructured interviews may include a brainstorming session or a formal interview amongst a group of people in their natural environment.

The major benefit of using unstructured interviews is that the interviewer is exposed to the perceptions of the interviewee and as Burns (2000:426) explains, that the corollary of this is the major disadvantage of unstructured interviews that is that the interviewer is being deprived of the unbiased truth. Therefore, if interviews the chosen vehicle for data collection, the choice of structured or unstructured seems to follow the choice of research perspective - with structured interviews being the positivistic choice and unstructured being the interpretive choice.

Cohen and Crabtree (2006) refer to the semi-structured interview as interviews where the interviewer sets up a general structure by deciding in advance the ground to be covered and the main questions to be asked. The detailed structure is left to be worked out during the interview, and the person being interviewed has a fair degree of freedom in what to talk about, how much to say, and how to express it. The advantage of the semi-structured interview is that it allows for a very flexible technique particularly for small scale research (like this project). Semi-structured interviews also allow informants the freedom to express their views in their own terms. This interview type can provide reliable, comparable qualitative data. The semi structured interview is not suitable for studies involving large numbers of participants

2.7.2 Observations

Adler and Adler (1994: 378) report a broad explanation of observation as “the act of noting a phenomenon, often with instruments, and recording it for scientific or other purposes”. Adler and Adler (1994:378) expand on this by explaining that observation can involve all five human senses either from direct interaction with the subject or hidden from the subject observing via recordings and other devices e.g. two way mirrors.

Adler and Adler (1994: 378) differentiate between quantitative observations and qualitative observations. Quantitative observations are designed for results to be repeatable and are therefore conducted under controlled settings whereas qualitative observation should occur in environments familiar to the subject which allow for natural influences.

Burns (2000: 405) lists four research stances for the participant observer:

1. “The complete participant” – this observer integrates oneself into the social context of the subject and operates under that social context, not revealing one’s identity and intention as an observer.
2. “The complete observer” – the subject of observation is unaware of the observer.
3. “The observer as participant” – The subject of observation is aware that the observation is taking place, however the observer is not integrated into the subjects’ social context.
4. “The participant as observer” - The subject of observation is aware that the observation is taking place, and the observer is integrated into the subjects social context.

As might have been intuitively understood, observation is used mainly within the ethnographic studies. As such, some of the advantages and disadvantages are quite similar to those of ethnography. The following lists some advantages of using observation:

- Allowing the researcher to make notes when it is pertinent from the researcher’s own perspective instead of relying on the subjects perspective (Burns, 2000: 411; Creswell, 2003: 186).
- Adler and Adler (1994: 382) emphasise the inconspicuousness of “the complete observer” as a unique way to gain insight into subject matter.
- Burns (2000:411) reasons that observation is necessary when the subjects are unable to speak e.g. infants, plants, animals.

The following is a list of some of the disadvantages or limitations of observation:

- Burns (2000:412) notes that qualitative observation is unpredictable because of external factors influencing the observation subject.
- Creswell (2004:186) lists some of the ethical considerations as limitations. These include private information and intrusiveness.
- There are difficulties in gaining a rapport with subjects e.g. animals and children (Creswell, 2003: 186).

2.7.3 This study

As discussed in Section 2.2.4, in order to measure how challenging a human participant perceives a task i.e. maintenance of systems or data warehouses, one needs to ask that participant. Thus, we need to employ interviews as a data collection method. This supports the elected research method, i.e. content analysis which deals with analysis of textual data. As we shall be dealing with interviews, and dealing with multiple participants' interpretation of the phenomenon and varying levels of relevancy, flexibility in what is transcribed shall be employed.

2.8 Conclusion

The goal of research is either to describe a phenomenon, predict its occurrence or understand the origin or cause of it. In this research, we are describing a phenomenon; that being "Challenges of maintenance and support." In describing this phenomenon, the goal is to establish the differences in the challenges between maintenance of transactional systems and the maintenance of data warehousing systems.

Research can be approached in a qualitative or quantitative manner. Quantitative research deals with hypothesis and experiments, making use of statistics to reach conclusions. Qualitative research deals with textual data and social interaction, making use of non-discrete, dynamic properties. Due to the phenomena being researched, this research is going to be approached in a qualitative fashion. There are three perspectives which influence qualitative research. These are positivist, interpretive and critical social theory perspective.

The positivistic perspective assumes that independence between researcher and participants. Positivistic research is objective. There is a hypotheses being tested and data is analysed using quantifiable measures of variables. Interpretivism assumes that the researcher is subjective as they interact within the social context of the participants. Knowledge is gained through understanding and analysis of this interaction.

Critical social theory research aims to change a society's incorrect interpretation of a phenomenon, especially where the incorrect interpretation a cause of oppression. This research deals with social interaction understanding a phenomenon from different social contexts and is thus interpretive.

Qualitative methods may take on a positivistic, interpretive or critical perspective. The methodologies discussed from a positivistic perspective were scientific method and surveys. The scientific method involves formulating a hypothesis and collecting data to support or refute the hypothesis. Surveys involve the use of data collection tools to collect data on a subset of a population. From the data collected on this subset, conclusions are drawn for the entire population.

The methodologies discussed from an interpretive perspective were ethnography, grounded theory and content analysis. Ethnography is a long term research method in which the research becomes part of the social context being researched. Ethnography employs observation as its main data collection tool. Grounded theory relies heavily on repetitive line by line coding and analysis of textual data. When using grounded theory, the research topic and themes should be discovered in the data as opposed to being tested for from the onset. Content analysis also involves coding however is more flexible than grounded theory because content analysis caters for the researcher's and participant's interpretation of data.

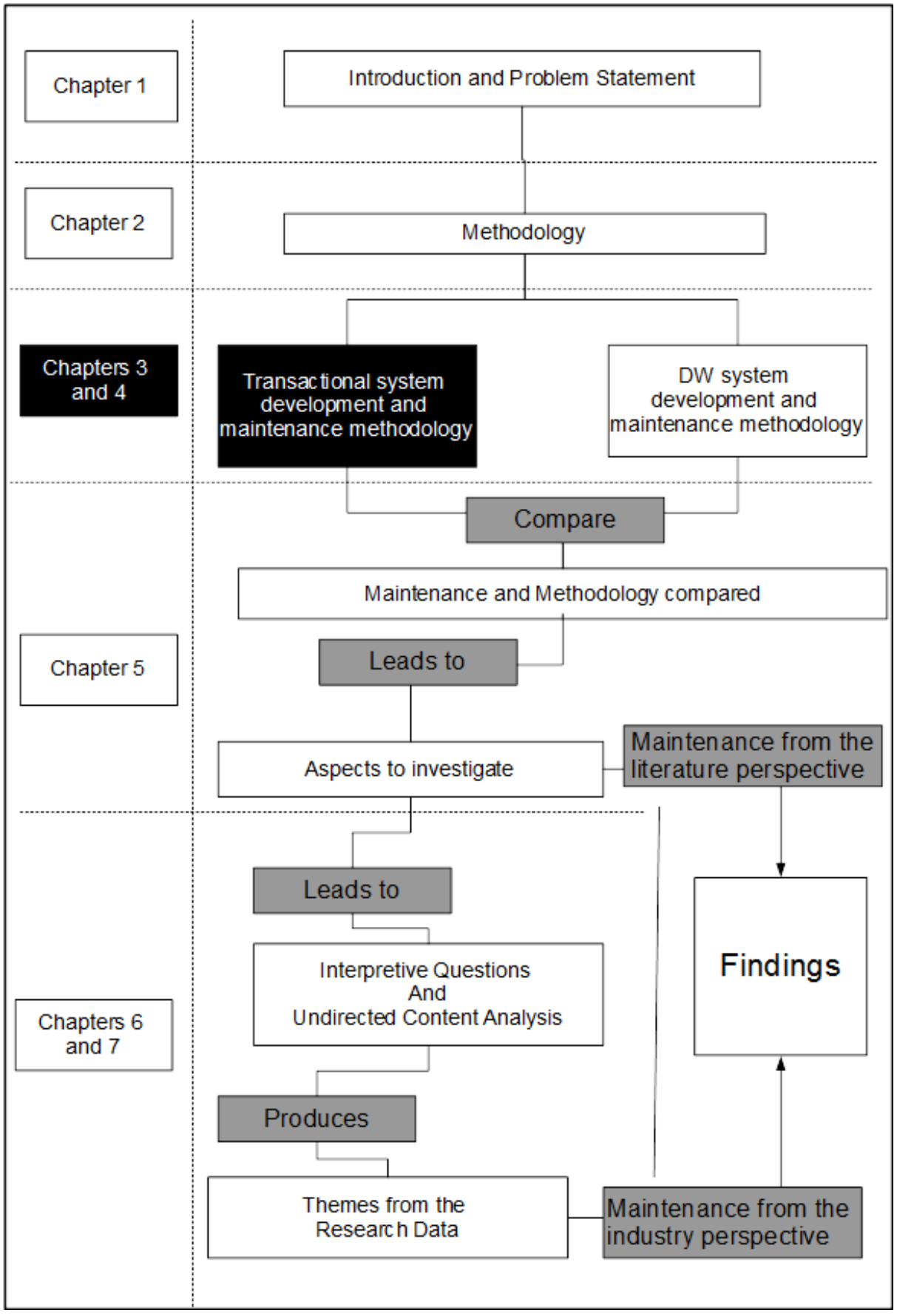
Action

research was discussed under the critical methodology. Action research first involves diagnoses of a problem and then involves a social process of rectifying the problem.

This study utilises the content analysis method as it lends itself to constructive and flexible analysis of textual data. The textual data is collected through transcribing interviews with software developers who have worked on data warehouse maintenance.

The details of the research, including the discussion on participant selection, interview questions, unit of analysis and coding scheme is discussed in Chapter 6. The coding scheme is of particular importance in validating findings and creating consistency amongst various differing interpretations of participants. The coding scheme is documented in a coding manual and included in the write up of the conclusions.

Prior to Chapter 6, a survey of the relevant literature on transactional software development methodologies is conducted in Chapter 3 and a survey on data warehouse development methodologies is conducted in Chapter 4. A review on transactional system and data warehouse system maintenance challenges is conducted in Chapter 5.


Chapter 1
Introduction and Problem Statement
Chapter 2
Methodology
Chapters 3 and 4
Transactional system development and maintenance methodology
DW system development and maintenance methodology
Compare
Maintenance and Methodology compared
Chapter 5
Leads to
Aspects to investigate
Maintenance from the literature perspective
Leads to
Findings
Interpretive Questions And Undirected Content Analysis
Chapters 6 and 7
Produces
Themes from the Research Data
Maintenance from the industry perspective

Chapter 3
Transactional Software Development and Maintenance

3.1 Introduction

The purpose of this chapter is to survey the methodologies used in transactional system development and examine how transactional systems are maintained. This is done in two major sections; software development models are examined in Section 3.2 and traditional software maintenance is examined in Section 3.3. This can be envisioned by examining the document organisational diagram on the previous page, and in particular, noticing the blocks shaded in black.

Boehm (2006: 12) documents the evolution of software development trends from the 1950s, starting by demonstrating that early software development models mirrored hardware development models. These were sequential lifecycles which naturally progressed into the extensively used "Waterfall Life cycle" (Boehm, 2006: 16; Huo *et al., 2*004: 2) during the 1970s. The trend of software development moved toward object oriented methods in 1980s. Concurrent processes and rapid application development in the 1990s evolved into agile methods in the 2000s. Going forward into the 2010s, there is a trend toward collaborative methods and global connectivity.

Some of the software development methodologies documented by Boehm (2006) are detailed in Section 3.2. These are the waterfall model in Section 3.2.1, software prototyping in Section 3.2.2, the spiral model in Section 3.2.3, object oriented programming in Section 3.2.4 and agile methodologies, specifically scrum in 3.2.5.

Maintenance of support software systems are detailed in Section 3.3. This begins with the standard maintenance process from the International Standards Organisation (ISO) in Section 3.3.1. Challenges in software maintenance are discussed in Section 3.3.2. While there may not be any single solution to address these challenges, means to reduce the challenge are offered in Section 3.3.3. These means include software evolution as a different perspective on software

maintenance and software tools as a means to take the burden off the software maintenance developer.

3.2 Software development models

Software development has been evolving since the 1950s when software development followed a similar methodology to hardware development. As much as each new methodology is similar to the previous one, there are noticeable changes which were effected to address challenges in development and maintenance. Some of the challenges are team centric such as in agile methodologies whilst others are business and risk centric such as the spiral model.

The following software development models are described in this section:

- the waterfall model in Section 3.2.1
- software prototyping in Section 3.2.2
- the spiral model in Section 3.2.3
- object oriented programming in Section 3.2.4
- agile methodologies, specifically scrum in 3.2.5.

3.2.1 The waterfall model

Even though the waterfall approach has become less popular, it is the most recognizable software development process (Huo *et al., 2*004: 2; Whitten *et al., 2*004: 40). Braude and Bernstein (2011:37) describe the waterfall process as being characterised by the sequential, ordered traversal through the phases depicted in Figure 3-1. Each phase is characterised by a pre-defined output, e.g. the output of the requirements phase is the requirements specification document.

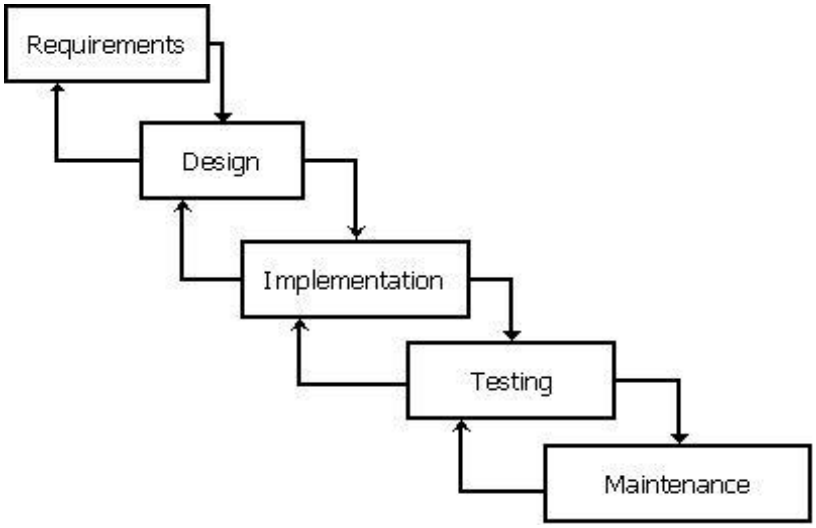


Figure 3 - 1 A typical waterfall process (B. Boehm, 2006: 15; Braude & Bernstein, 2011: 37)

One might notice the loopback arrows in Figure 3-1 flowing up the waterfall. Braude and Bernstein (2011: 37) explain that while the process is theoretically sequential, there are frequently send-backs and re-visits to previous steps which emerge in real life.

Lewis (2009: 87) summarises the first phase, requirements, by stating that users are interviewed and these interviews are collectively analysed to produce the requirements document. Developers get involved from the design phase. Lewis (2009:87) names entity relationship diagrams, process decomposition diagrams and data flow diagrams as some of the logical design outputs and goals of this phase.

Whitten *et al.* (2004:39) express that new hardware as well as software is installed during the implementation phase. If the software is custom built, this is the phase where the coding occurs. The testing phase consists of testing the complete system as well as integration with other interfacing systems. This is deemed a limitation of the waterfall process by Braude and Bernstein (2011: 38) as it is as very late to be looking at the possibility of design changes.

System maintenance and support takes the form of a sub-project which starts again at the requirements gathering phase where the requirements are usually bug/error reports and/or enhancement requests from users (Whitten, *et al.* 2004: 40).

Advantages of waterfall model

Braude and Bernstein (2011: 38) discuss some of the advantages of the waterfall model:

- As it has been in practice for a long time, it is widely understood. This makes it easier to use as everyone involved has an idea of the stages involved and the sequence in which these are executed.
- The sequential nature of the model facilitates understanding and management. As each stage's completion is a prerequisite for the next stage's inception, timelines are easy to plan. Differently skilled resources are also easier to plan for as one can estimate when the phases requiring these skills shall begin.

Disadvantages of the waterfall model

- Pressman (2001: 38) warns that requirements are collected in the beginning and there is little flexibility to change these requirements. This disadvantage is further compounded because, as Braude and Bernstein (2011: 38) explain, many problems are discovered only in the testing phase. From project perspective, one expects to be almost complete with a project in the testing phase and spending time repairing bugs and errors are often not catered for.
- The sequential nature of the process prevents disjointed tasks from being executed at the same time. Braude and Bernstein (2011: 38) reason that this leads to some team members being idle whilst waiting for different phases to complete.
- Due to this inefficient use of time and resources, the waterfall process can be construed as being wasteful and expensive.

3.2.2 Software prototyping

Prototyping involves building preliminary versions of parts of a product to permit early feedback from users and stakeholders (IEEE Std 610.121990, 1990: 60). Doing this iteratively could lead to the prototype becoming operational (Boehm *et al., 1*984: 302).

The goal of prototyping, summarised by Hudak and Jones (1994: 1), is to better understand the problem and solution thereby reducing risk. This is because, as explained by Braude and Bernstein (2011: 40) we usually do not understand all factors at the beginning of a project. Hence prototyping can be used to get a better understanding of requirements.

This improved understanding of requirements is acquired by putting together a quick design; focusing on aspects of the user interface (Pressman, 2001: 30). This user interface is then enhanced into a prototype which the customer evaluates. The process of enhancement is an iterative one which typically runs in parallel to the regular work on the project. While the prototype focuses on the user interface, work may continue on data modelling and development.

After a few iterations of enhancement, the prototype may include data access and become more functional. At this point, Braude and Bernstein (2011: 41) emphasise that that the prototype development should focus on risk mitigation by ensuring that the highest priority risks are incorporated into the prototype development thereby allowing the customer to evaluate these risks early on. This is advantageous because “studies have shown that the cost to fix a fault early in the software life-cycle is exponentially less than the cost to fix it later” (Hudak & Jones, 1994: 1).

<u>Advantages of prototyping</u>

- Braude and Bernstein (2011: 44) recommend using prototyping when doing feasibility studies as it is an ideal way to assess risk and feasibility.

- “Both customers and developers like the prototyping paradigm. Users get a feel for the actual system and developers get to build something immediately” (Pressman, 2001: 31).

Disadvantages of prototyping

- Often prototyping requires real-life simulations which can be difficult, expensive and time consuming to create (Braude & Bernstein, 2011: 44).
- Pressman (2001:32) explains that prototypes are put together without concern for longevity and performance of the system. If this type of prototype evolves into a production system, it could spell disaster due to unforeseen bugs and performance issues.

3.2.3 The spiral model

Essentially, the spiral model is an iterative evolutionary risk driven process (Braude & Bernstein, 2011: 44; Pressman, 2001: 37). The goal of each iteration is to reduce risk; whilst increasing the scale and usability of the development. Pressman (2001: 37) simplified Boehm’s (1988: 64) spiral model as depicted in Figure 3-2.

It is important to note the “project entry axis” in Figure 3-2. Each entry point (*viz*. develop concepts, new project, enhance and maintain) represents a theoretical iteration of the spiral model. In practice, Pressman (2001:38) explains there are usually multiple iterations around the spiral prior to reaching the next phase on the project entry axis.

During the first phase on the project entry axis of the spiral model, communication with the customer results in objectives being set and concepts being developed. Boehm (1988:65) emphasises that multiple concepts are evaluated and the risk for each one is assessed. This may take the form of evaluating off the shelf solutions and weighing them against possibilities that a custom built solution might offer. Pressman (2001:37) speculates that the first iteration usually results in a product specification.

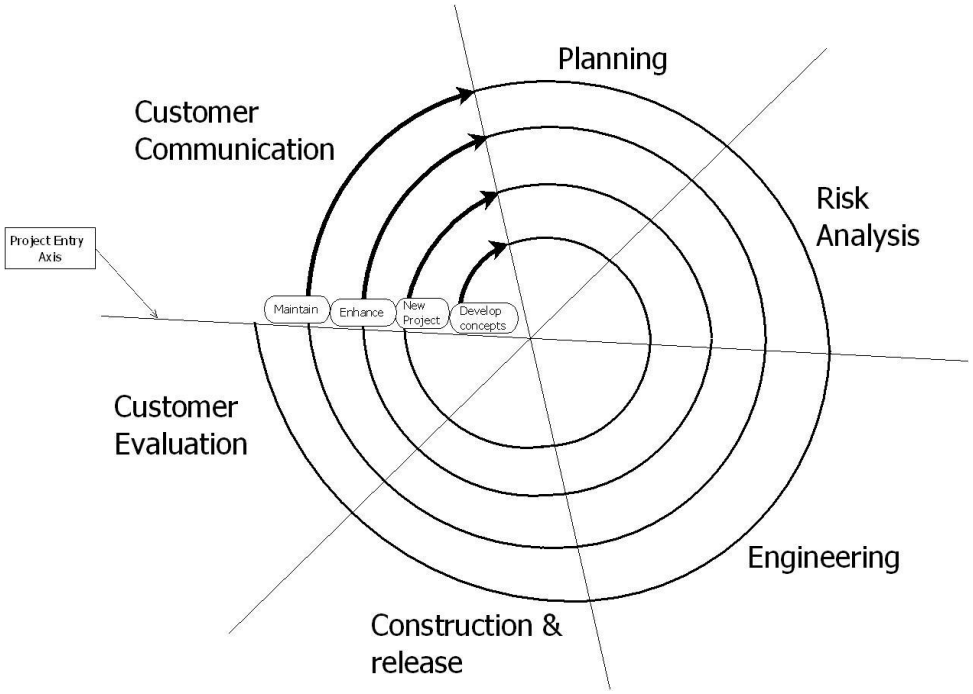


Figure 3 - 2 The Spiral Model (Pressman, 2001: 37)

During the next iteration, the customer works with the developers to build on the output of the previous phase by giving input regarding further requirements and analysis. Boehm (1988:65) proposes that this iteration could result in continuous prototyping and risk evaluation of each prototype produced if prototyping is a viable option. For larger projects, or where prototyping has fulfilled its objectives, Boehm (1988:65) proposes that the "new project" phase/iteration in Figure 3-2 shall follow the waterfall approach, "modified as appropriate to incorporate incremental development" (Boehm, 1988: 65). If a modified waterfall approach is taken, the modifications made to the waterfall process should entail analysis and verification at the end of each phase in order to reduce risk.

Once the project has been delivered, enhancement and maintenance follow a similar iterative process through the spiral. Typically, the user requests an enhancement. After planning and investigation, the risk is evaluated and work commences. Initially, quick prototypes of the enhancement are evaluated to reduce the risk of developing a fully functional enhancement that is not appropriate. This evolves into an acceptable end product.

Advantages of the spiral model

- Boehm (1988:69) emphasises that the distinctive advantage of the spiral model is the ability to incorporate other software process models into the model whilst reducing the risk associated with these models.
- The spiral model allows software to evolve as the project progresses because the iterative nature reassesses risk and requirements continuously (Braude & Bernstein, 2011: 46; Pressman, 2001: 38). This makes it a more realistic approach to large-scale development than the classic waterfall model.

Disadvantages of the spiral model

- Boehm (1988: 70) cautions against the utilisation of the spiral model when software development is outsourced. This is because contracts are generally bound by documented deliverables prior to the onset development on whereas the spiral model relies on constant communication with business in order to evaluate risk during the process of development. This leads to deliverables being constantly re-estimated.
- Braude and Bernstein (2011: 46) point out that risk analysis is a specialised field which makes the use of the spiral model complicated.

3.2.4 Object oriented programming

Before discussing object oriented design, it is pertinent to draw a distinction between the object oriented analysis and design development methodology and object oriented programming. Whilst many terms are common to the style of programming and the software development methodology, these two are distinct topics. The topic being discussed here is the methodology. An in-depth discussion of object oriented programming is out of the scope of this dissertation. However, some basic components which may be applicable to both the methodology and the programming style need to be introduced.

The basis of object orientated design is the concept of the object. “An object is an abstract software model of a problem domain entity. Objects are packages of both

data and operations on that data" (Seidewitz & Stark, 1987: 54). "The term class is used to describe a set of objects with similar behaviour…a class is also used as syntactic mechanism in almost all object-oriented languages." (Budd, 1997: 40)

Classes and objects are reusable and customisable without having to duplicate code because of object oriented concepts such as inheritance and polymorphism. Inheritance as described by Budd (1997: 12) is the grouping of objects into a singly rooted tree structure which allows attributes and functionality of classes higher up in the tree to be automatically available to classes lower down. Budd (1997: 252) explains that polymorphism allows a class to be tailored to fit specific circumstances by replacing certain attributes, functions or methods with a specialised version of them whilst being able to use the rest of the class as the original.

Having provided an introduction to key concepts, attention can now be given to object oriented design. This is defined as "a software development technique in which a system or component is expressed in terms of objects and connections between those objects." (IEEE Std 610.121990, 1990: 51).

Object oriented design entails the following:

" 1. Identify key problem-domain entities
2. Distinguish between active and passive entities.
3. Establish data flow between active entities.
4. Decompose entities (or functions) into sub-entities and/or functions.
5. Check for new entities.
6. Group functions under new entities.
7. Assign new entities to appropriate domains " (Bailin, 1989: 609).

Whilst Bailin (1989) refers to the above as "an object-oriented requirements specification method" and Bailin (1989: 621) does recommend further steps to transition into a design; these steps do include "defining abstract data types" and "reusable functions". It is because of these inclusions that Hendersen-Sellers and

Edwards (1990: 149) point out that steps 4 to 7 can be linked directly with the analysis, design, and implementation stages.

Bailin's (1989: 610) recommended method for identifying objects and attributes is to locate nouns and noun phrases in the requirements documentation. As part of this step of identifying objects as well as to streamline the next step of identifying operations on the object, Bailin also distinguishes between active and passive objects (or entities as referred to by Bailin). "An active entity is one that operates on inputs to produce outputs. A passive entity is one that is acted upon" (Bailin, 1989: 611).

Using these definitions of active and passive objects, an entity relationship diagram (ERD) is constructed which identifies relationships between objects. Thereafter, a data flow diagram (DFD) is constructed in which operations affecting these objects are identified for analysis (Bailin, 1989: 612; Fichman & Kemerer, 1992: 26; Henderson-Sellers & Edwards, 1990: 149) .

Fichman and Kemerer (1992: 26) recommend that steps 4 to 7 of Bailin's (1989) method are to be performed iteratively. Larger entities are broken down into simpler variants in step 4. This may lead to definitions of new entities in step 5. Step 6 involves identifying functions performed by the new entities and re-organisation of the data flow diagrams. Fichman and Kemerer (1992: 26) explain that these entities and functions are then grouped into different domains of the application in step 7. An ERD and DFD are then created for each of these domains.

<u>Advantages of object oriented design</u>

- As many programming languages are geared towards object oriented programming, Booch (1994: 284) points out that object oriented design encourages the methodology itself to exploit the full power of these languages. This includes the reuse of software components, and building software that is more resilient to change through inheritance and polymorphism.

- Individual objects and modules of a system can be designed in isolation as and later integrated into a system (Korson & McGregor, 1990: 52). This is because all objects share the same kind of interface, which is a class.
- Object orientation is easier to reverse engineer for re-use of common designs. Nerson (1992: 72) explains that this is because all objects are modelled into data structures which translate back into an ERD schema.
- Most object oriented languages allow for information hiding, or encapsulation by separation of the class interface and implementation (Korson & McGregor, 1990: 51).

Disadvantages of object oriented design

- When dealing with very large systems, Fichman and Kemerer (1992: 37) point out that conventional methodology provide more intuitive mechanisms to divide the these larger systems into more manageable parts for multiple team development. As object oriented programming deals with objects on a very low level, and these objects are often highly interlinked, it becomes challenging to divide system development into multiple teams.
- Booch (1994: 284) warns that object oriented programming could lead to performance degradation. Some of the reasons for this are:
    - When objects' methods invoke each other, a dynamic lookup is needed when data types are not predefined. This is slower than static lookups.
    - Object oriented programming has multiple layers of abstraction due to encapsulation and inheritance. The implication of this is that multiple method calls are required as higher level methods build on lower level methods. Booch (1994: 284) reasons that this cascading of method calls would have adverse effects on performance.

3.2.5 Agile methodologies: Scrum

There are a number of methodologies that fall under the Agile category of methodologies (Hayes & Andrews, 2001: 8). Scrum, extreme programming (XP), Crystal and Kanban are names of some of the more common ones. To cover all of these methodologies would be beyond the scope of this research as well as

unnecessarily repetitive. Therefore, the most commonly used agile methodology was chosen to be discussed. Begel and Nagappan (2007: 255) have found scrum to be the most commonly used agile methodology by Microsoft Developers. Another study, sponsored by Version One (2010) also found scrum to the most commonly used amongst the agile methodologies.

Agile methodologies all have different methods to develop software; however they all follow the same of principles which govern the Agile Manifesto (The Agile Alliance, 2001). In summary, these are:

- To satisfy the customer through early and continuous delivery of software.
- To be adaptive and welcome changing requirements, even late in the development.
- To use short release and development intervals.
- To be team driven and team oriented.
- To ensure that business is included in the software development process.
- To have constant face-to-face communication with the team and with business.

The rest of this section delves into the particulars of scrum.

3.2.5.1 Introduction to scrum

Scrum is described as a framework rather than a methodology (Cohn, 2011a; Sutherland, Downey, & Granvik, 2009: 69). Scrum defines a short development iteration cycle called sprints. The duration of these are usually two weeks to a month. During a sprint, the team tries to complete a subset of work defined from the larger set of prioritized requirements from business. Kniberg (2007: 9) emphasises that this larger set of requirements, known as the product backlog is the central, dynamic driving force of work. The people involved in this framework, as listed in are (Cohn, 2011a) are:

- A self-organised scrum team. This is a development team without a designated leader as decisions are taken by the team as a whole.

- The scrum master. This is a member of the scrum team that assists the team with the implementation of scrum by providing advice on amount of work being committed to and a facilitator of daily meetings.
- The product owner. Hayata and Han (2011: 286) identify this role as being typically filled by someone from business who is responsible for communicating the highest priorities from the product backlog.

“Meetings are the heart of scrum and represent agile methods that promote project flexibility and speed. Scrum has four types of meetings: kick-off meeting, sprint-planning meeting, daily-status meeting, and sprint-review meeting” (Hayata & Han, 2011: 287). Cohn (2008: 13) also lists a short sprint retrospective; however this may be included in the sprint review. These meetings are the major activities defined in the scrum framework and are discussed in detail in the sections to follow.

3.2.5.2 The kick-off meeting

Hayata and Han (2011: 287) impart that the goal of the kick-off meeting is introduce the new scrum team as well as identify the product owner and create the product backlog.

With scrum, requirements generally unfold incrementally (Soni, 2007) therefore the product backlog is a dynamic requirements document. Kniberg (2007: 9) explains that the initial high level requirements are set out in the definition of a product backlog. This is made up of a high-level prioritized list of requirements which are called stories. These stories will later be segregated into smaller units of work. The product backlog is owned by the product owner hence the stories should be in a form which contains the business reason for the stories. Kniberg (2007: 12) reasons that this ensures that technical details do not overshadow business goals.

3.2.5.3 Sprint planning meeting

In scrum, work is done in short intervals, usually about 2 to 3 weeks in length. These intervals are called sprints. Sprints begin with a sprint planning meeting which as explained in (Cohn, 2011b), includes the entire scrum team, including the product

owner. During this meeting, the most important stories from the product backlog are discussed by the product owner and the team decides on which of these important stories can be included in the forthcoming sprint.

The entire development team is involved in this making this decision by using a formalised process known as planning poker. This process allocates a score, called story points to each of these important backlog stories. Story points generally correlate to estimated time units, either working days or hours, involved to complete the story. These story points are then used to determine how much of work to attempt during a sprint period. The term used for this amount of work is velocity.

3.2.5.4 Daily status meeting

These are daily short team meetings where each member of the team quickly summarises the work that they have been working on the previous day, the work that shall be worked on in the current day and any obstacles that are hindering progress with that work (Cohn, 2004: 171). Hayata and Han (2011: 287) refer to these meetings as “status meetings”, “scrums”, or “standups”. Members from business may attend these meetings as observers as this might give an idea of progress of the work involved. It is important to keep these meetings short e.g. 15 minutes. A need to discuss obstacles at length may arise out of the daily scrum and this need should be catered for in a separate meeting attended by interested and affected individuals.

Daily scrums occur while referring to a sprint task board as in Table 3-1. This board contains the stories agreed on during the sprint planning meeting. Stories are placed on the board from top to bottom in the order of business importance which Kniberg (2007: 53) explains is the order in which they are worked on. Tasks in the “Awaiting Test” and “To do” column should be picked up by the first available team member, who should prioritise working on tasks in higher up columns.

Story	To Do	In Progress	Awaiting Test	Test in Progress	Done
STORY A: (Description) 8 points		Task A4 Task A6 Task A7 Task A8 Task A9	Task A2 Task A5	Task A3	Task A1
STORY B: (Description) 5 points		Task B4 Task B5 Task B6 Task B7	Task B1	Task B3	Task B2
STORY C: (Description) 13 points	Task C5 Task C6 Task C7	Task C4 Task C8	Task C1	Task C3	Task C2

Table 3-1: Template for a sprint task board ("Training for scrum Task Board Use," 2011; Kniberg, 2007: 53)

Progress of the sprint is tracked at daily scrums by using a burn down chart (Kniberg, 2007: 51; MSDN, 2010). This is a chart which plots total story points on the vertical axis and number of sprint days on the horizontal access. During daily scrums, completed stories have their story points subtracted from the total giving the chart a negative gradient. A quick glance at the burn down chart allows the scrum master and rest of the team to assess the progress of work during the sprint. Progress is discussed in greater detail during the sprint review meeting.

3.2.5.5 Sprint review and retrospective meetings

Hayata and Han (2011: 286) summarise the activities of the sprint-review meeting as being where the work done during the sprint is reviewed and demonstrated to stakeholders. It is important to assess whether the sprint goal was achieved and also assess the stories that were not completed (Cohn, 2011d).

Another meeting at the end of a sprint is the sprint retrospective. Knieberg (2007:68) highlights the purpose of this meeting as making improvements for the next sprint. The sprint burn down chart comes into play and the difference between estimated velocity and velocity achieved are examined. Each team member contributes to the meeting by writing a note on what was done well during the sprint, what should have been done better and what the major obstacles were. These notes are put up on a board within aptly name columns e.g. “good, bad, obstacles”. Improvements to be made for the next sprint are decided on through voting on these notes. At each retrospective, attention is given to the notes voted on during the prior retrospective.

3.2.5.6 Advantages of scrum

- Overhage *et al.* (2010:3) report that scrum teams can ensure early that requirements are meeting expectations from the customer as there is frequent communication and the team demonstrates the software incrementally.
- Schwaber and Beedle (2002: 109) reason that scrum reduces the risk of not completing important functionality by planning development according to business importance.
- Because of the emphasis that scrum puts on face communication, ideas are transferred and understood more effectively than written communication (Highsmith & Cockburn, 2001: 121).
- Cockburn and Highsmith (2001: 133) report that agile teams have better morale.
- Because of daily scrum meetings, progress is tracked regularly. This lowers the risk of poor estimating and planning (Schwaber & Beedle, 2002: 109).
- By design, agile methods are more adaptable and welcoming of changing requirements. This adds flexibility to the software development process (The Agile Alliance, 2001).

3.2.5.7 Disadvantages of scrum

- Boehm (2002: 65) warns that while team-oriented development has many benefits, it could lead to too much input from less experienced and qualified individuals.
- The customer in the form of the product owner is very involved in the development process. Some customers may view this as time-consuming and bothersome (Overhage *et al., 2*011: 3).
- When setting up infrastructure and architecture, Boehm (2002: 66) explains requirements need to be known early on as customers need to keep hardware acquisition within budget.
- Cockburn and Highsmith (2001: 133) reason that larger teams will most likely find agile projects difficult.
- Overhage *et al.* (2010:4) put forward concerns that the lack of formal documentation may leave a knowledge-sharing gap when key team players leave.

3.3 Software maintenance

Swanson (1976: 494) lists three basic types of software maintenance. These are corrective, adaptive and perfective maintenance. This list has since been expanded by the International Standards Organisation and the International Electrotechnical Commission (ISO/IEC) as well as Institute of Electrical and Electronics Engineers (IEEE) in ISO/IEC/IEEE standard (Std) 14764 (2006: 3) to include emergency maintenance and preventive maintenance.

- Corrective maintenance is associated with the processing failure of software. Swanson (1976: 494) explains that processing failures occur when the software does not process inputs as expected resulting in erroneous or incorrect outputs.
- Emergency maintenance is part of corrective maintenance, however ISO/IEC 14764 (2006: 3) asserts that it is not scheduled.
- Preventive maintenance is viewed as being a subset of perfective maintenance by Chapin (2000: 16) and Kajko-Mattsson (2000: 12). Both these authors feel that the ISO/IEC Std 14764 (2006: 4) definition: “the

modification of a software product after delivery to detect and correct latent faults in the software product before they become operational faults" is unclear. Chapin (2000: 16) suggests that the term preventive maintenance could be used to describe scheduled maintenance done with the aim of reducing maintenance costs and improving quality.

- Perfective maintenance: ISO/IEC and IEEE Std 14764 (2006: 4) distinguish perfective maintenance from operational maintenance in that it is done to prevent latent faults from becoming failures. Examples provided are "enhancements for users, improvement of program, documentation, and recoding to improve software performance, maintainability, or other software, attributes" (ISO/IEC 14764, 2006: 4). Swanson (1976: 494) summarises that perfective maintenance is that which is done to improve processing efficiencies and enhance performance.
- Adaptive maintenance: Swanson (1976: 494) explains that this is maintenance performed in response to changes in the environment. This might be business driven changes e.g. additions to product classifications or technology driven changes e.g. a newer version of a database management system.

3.3.1 The software maintenance process in the software life cycle process

Software development methodologies each approach maintenance in accordance to the underlying principles of the methodology e.g. sequential, iterative or *ad hoc*. In order to gain a thorough understanding of activities undertaken during maintenance and the challenges of these, one needs a more in-depth understanding of maintenance. However, to discuss all methodologies' maintenance phase in-depth would be very repetitive. Therefore, a discussion that follows refers mainly to the ISO standard on maintenance.

ISO/IEC 14764 (2006: 5) defines the maintenance process as part of the life cycle process. A process for maintenance should be created prior to the development occurring. The standard defines the objective of the maintenance process as

modifying the existing software product while preserving its integrity. The activities involved in the maintenance process are shown in Figure 3-3 below. Whilst the process is iterative, the activities are numbered according to the order in which they initially occur.

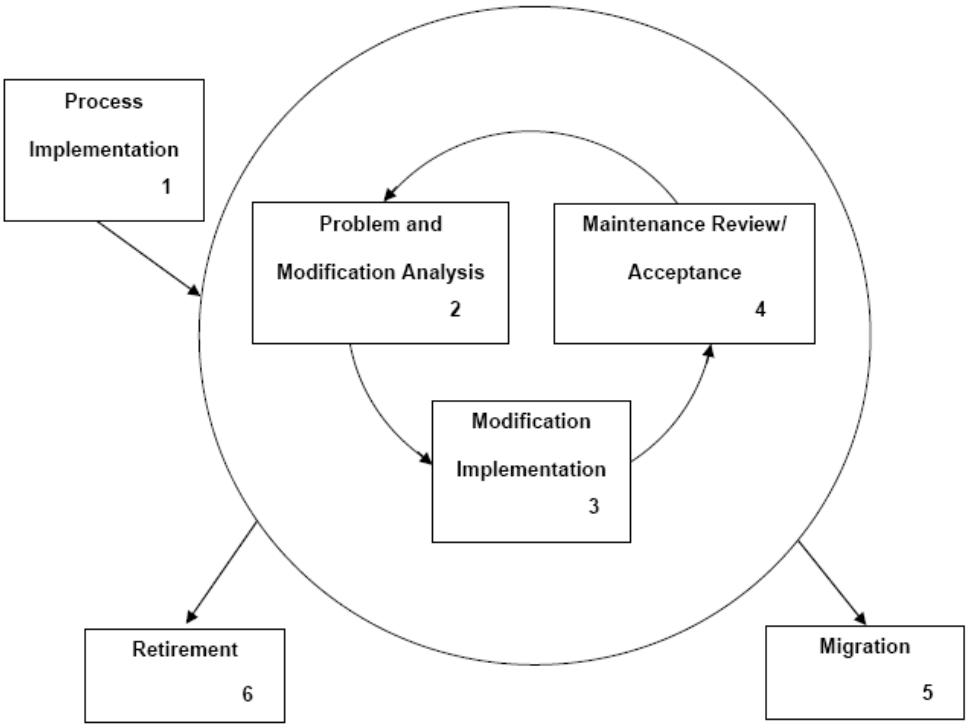


Figure 3 - 3 Activities in the maintenance process (ISO/IEC 14764, 2006: 6)

The activities (2, 3 and 4) within the circle in Figure 3-3 form part of the work done during the maintenance of software. Activity 1, implementation is an input to the work being done because it establishes the maintenance process (ISO/IEC 14764, 2006: 6). Activity 5, migration occurs when maintenance is done to migrate a system to a different environment. The migration activity signifies the end of the current maintenance process as a new process should be created for the new environment. A software retirement activity is included in the maintenance process by ISO/IEC 14764 (2006: 24). Software retirement also signifies a conclusion of software maintenance.

3.3.1.1 Activity 1: Process implementation

A strategy for performing maintenance is developed to ensure that that maintenance is successfully implemented.

The tasks involved to accomplish this are:

- Developing maintenance plans and procedures. ISO/IEC 14764 (2006: 7) list the steps involved as defining the scope of maintenance, estimating costs by conducting resources analysis and evaluating maintenance organisation alternatives and identifying and documenting the maintenance process.
- Establishing maintenance request (MR) and problem report (PR) procedures. A procedure for submitting an MR/PR should be developed which includes a numbering, prioritising and categorising scheme. In addition to this, ISO/IEC 14764 (2006: 8) specifies that procedures be developed for: determining and analysing maintenance trends, receiving and providing feedback from users and providing work-arounds.
- Develop a configuration management plan and process (ISO/IEC 14764, 2006: 7). Maintenance, by definition involves modification of software. Configuration management involves managing these modifications. ISO/IEC 12207 (2007: 68) defines configuration management process as including a means of requesting, approving, denying and analysing change requests. Details of the implementation, verification and release of the modified software item must also be captured. All of this information must be captured traceable and auditable.

3.3.1.2 Activity 2: Problem and modification analysis

Before modifying a system, the MR/PR needs to be analysed. ISO/IEC 12207 (2006: 9) prescribes that this includes an initial impact analysis of the handling the MR/PR, verification of the problem, consideration of options available to perform maintenance for MR/PR, documentation of analysis done and finally obtention of approval for the work to be done.

The purpose of doing an analysis of the MR/PR is to understand the impact of attending to the MR/PR, classify the maintenance and prioritise the handling of the MR/PR (ISO/IEC 12207, 2007: 54; ISO/IEC 14764, 2006: 10). It is pertinent to accomplish this prior to continuing with any further analysis. Many factors are taken into account to understand the impact of a modification. Some of these are budget, staff availability, infrastructure capacity, impact on the organisation, existing system and interfacing systems.

The verification of a problem is not always straightforward. A problem could occur under very specific circumstances and being able to replicate the problem sometimes requires replication of the circumstances. ISO/IEC 14764 (ISO/IEC 14764, 2006: 11) summarise the steps involved in problem verification as: developing a test strategy for the problem, obtainment and installation of the necessary software and data components, running the test and documenting the test results. Problem verification is only required for corrective maintenance.

Once the problem has been verified and assigned a priority and classification, options for maintenance implementation should be considered (ISO/IEC 12207, 2007: 54; ISO/IEC 14764, 2006: 11). A risk analysis of each option should be undertaken and documented. The outcome of this step is an agreed-upon plan to implement

Documentation should be ongoing, however it will need to be reviewed as it will be required when requesting approval. Obtainment of approval involves analysis of documented results and discussion with change and configuration management groups.

3.3.1.3 Activity 3: Modification implementation

The modification should be done according the team's tailored development standards (ISO/IEC 14764, 2006: 14), however ISO/IEC 12207 (2007: 54) does stipulate that the following be added to the requirements of these standards:

“ a) Test and evaluation criteria for testing and evaluating the modified and the un-modified parts software units, components, and configuration items) of the system shall be defined and documented.

b) The complete and correct implementation of the new and modified requirements shall be ensured. It also shall be ensured that the original, unmodified requirements were not affected. The test results shall be documented.”

3.3.1.4 Activity 4: Maintenance review/acceptance

This step involves reviewing the modifications to ensure correctness and utilisation of the approved standards and methodology. Approval of the modification is granted for a satisfactory review (ISO/IEC 14764, 2006: 15).

ISO/IEC 14764 (2006: 14) details many task steps for reviewing the modification. These include verification of code and coding standards, verification that only necessary components were modified and that these are properly integrated, and verification that the documentation has been updated. Component and full system testing should be performed by change management and an independent test organisation.

Approval involves following a configuration management process. This includes auditing the change, notifying users, installation and training (ISO/IEC 14764, 2006: 15).

3.3.1.5 Activity 5: Software migration

When a requirement exists to migrate a system to a new environment a migration plan should be developed, documented and executed.

Users should be included in the plan development. ISO/IEC 12207 (2007: 55) includes the following in migration plan development: requirements analysis and definition of migration, migration tool development, data and software product

conversion, migration implementation and verification and future-support for the old environment.

Communication to users need to be sent out informing them of migration plans and activities, including a reason for the new system, its date of availability and what (if any) type of support will be available for the old system (ISO/IEC 12207, 2007: 55).

Both new and old systems may run concurrently to allow for a smooth transition. ISO/IEC12207 (2007: 55) recommends that this transition period also be used for training on the new system. At a scheduled date, user access to the old system is revoked and notification should be sent out. All old environment documentation and data should be archived and accessible in accordance with requirements for data protection and being auditable. The impact of the migration should be assessed in a post-operation review.

3.3.1.6 Activity 6: Software retirement

This is a process which ends a software component's existence. ISO/IEC 12207(2007: 56) notes that this process is necessary to ensure that the integrity of the operation of the organisation is unaffected.

As with most processes, the first step is to create a software retirement plan. The creation of the plan includes an analysis of the retirement requirements and impact of retiring the software product (ISO/IEC 14764, 2006: 22). Any replacement product should be identified and a schedule of cessation of the product and product support should be established to coincide with transition to a new product, if applicable (ISO/IEC 12207, 2007: 56). The retirement effect should be documented and any artefacts should be controlled by configuration management.

On the scheduled retirement date, notification of the retirement must be dispatched to all concerned. ISO/IEC 12207 (2007: 57) reiterate that the data associated with

the software component being retired should be archived and available on request for audit and legal requirements.

3.3.2 Challenges in software maintenance

It is widely reported that maintenance is the costliest stage of software development, either in terms of developer's time (Lientz, 1983: 272), or in terms of total life cycle costs (Polo *et al., 1*999: 76), or in terms of reworking developed software (Boehm & Papaccio, 1988:1466). The act of implementing software maintenance itself compounds the cost factor as it is often associated with business down-time (Mookerjee, 2005: 76; Xiong, Xie, & Ng, 2011: 25).

Despite this huge cost factor, Polo *et al.* (1999: 76) report that not much attention has been given to implementing maintenance methodologies such as the ISO process covered in Section 3.3.1. This could be seen as a lack of initiative or understanding from managers or executives who tend to focus on short term pressures and neglect to plan for preventative and/or perfective maintenance. This type of short term focus could lead to software failures (Lehman, 1980: 1066; Lientz, 1983: 274; Stachour & Collier-Brown, 2009: 54). Mens *et al.* (2005: 18) emphasises the need to increase the awareness of executives of the importance of maintaining software.

One consequence, reasons Lientz (1983: 274), of not following a maintenance methodology and focusing on short term goals is that systems are documented hastily and incompletely. Lethbridge *et al.* (2003: 36) have found that software engineers are unlikely to update documentation timeously and thoroughly unless compelled to do so as part of the change process yet documentation was found to be useful and valuable by software engineers.

Selic (2009: 11) warns that poorly documented systems lead to greater challenges and costs during maintenance. Nguyen *et al.* (2010: 686) have found locating and understanding code segments requires the greatest portion of development time

when doing corrective maintenance. System experts and senior team members may reduce the challenge by instructing and mentoring however this too could prove inefficient and frustrating as these experts and senior team members could find themselves having to repeat information that ought to be documented to different groups (Selic, 2009: 11).

The effect of manager's lack of interest in perfective/preventative maintenance is often compounded by the challenge of users' fervent appetite for enhancements and extensions to a current system. Lientz and Swanson (1981: 769) report that these user demands are seen as the most severe overall maintenance problem. Lientz and Swanson (1981: 769) also mention that this problem may be related to lack of user understanding of the system and inadequate training.

The findings of Lientz and Swanson (1981: 765) show a significant correlation between system size and problems associated with maintenance. Unfortunately, systems have been getting much larger since then and will continue to grow (Xiong *et al., 2*011: 21). Individual systems become larger and even more difficult to maintain when integrated with other systems. Mookerjee (2005: 75) reports that such system integration is a growing trend. Mens *et al.* (2005: 4) reasons that integrating these systems often requires working with multiple programming languages which might pose a developer skill and human resource (HR) challenge.

The findings of Niederman *et al.* (1991: 480) show that information system resource recruitment is a high ranking management issue due to requirements for continuously increasing and specialised skills. This issue runs the risk of being escalated by downsizing policies during economic recessions. Lientz (1983: 271) warns that high turnover of software maintenance personnel may lead to reduced support of application systems or even system damage due to erroneous enhancement or fixes. More recently, Ramanujan *et al.* (2000: 1151) have found that experienced programmers required less effort to carry out complex maintenance tasks than inexperienced programmers and yet according to Selic (2009: 11), maintenance is usually delegated to junior staff.

Some companies are still facing the challenge of maintaining legacy systems which Dedeke (2012: 38) defines as “an aggregate package of software and hardware solutions whose languages, standards, codes, and technologies belong to a prior generation or era of innovation.” Legacy systems are due to software aging. Lehman (1980: 1068) aptly describes the process of software aging: “as it ages, those working on or with the system become increasingly constrained by earlier decisions, by existing code, by established practices and habits of users and implementers alike”.

Bennet (1995: 20) reviews the maintenance challenges of legacy systems as being: the usage of outdated technologies and coding strategies, lack of developer interest and shortage of skills. Whilst these systems are very costly to maintain, Dedeke (2012: 38) explains that they are often still operational due to cost of replacement, uncertainty of change and slow adoption of new technologies. Due to these conflicting factors, maintaining legacy systems usually means dealing with the dilemmas of software migration and/or software retirement (Bennett, 1995: 20; Dedeke, 2012: 39)

Commercially available off-the-shelf (COTS) systems are being used extensively due to perceived cost saving and perceived ease of implementation however these perceptions are often false (Basili & Boehm, 2001: 91). Over the long term life cycle of the system, maintenance challenges of COTS systems include: more specialised developers, more frequent updates and upgrades from the vendors which could translate into more system downtime, higher system integration costs, more maintenance developer time and effort, and a higher risk of losing software support due to outdated product versions (Reifer, Basili, Boehm, & Clark, 2003: 94; Kotony & Hutchinson, 2005: 69; Basili & Boehm, 2001: 91).

Outsourcing development and maintenance is being used to deal with the lack of skilled resources available. Ahmed (2006: 450) cautions that outsourcing could lead to poor quality software and documentation with lack of accountability from vendors.

This could lead to added maintenance effort and costs in rewriting software. Campbell (1995: 22) warns that outsourcing maintenance allows outsourcing vendors to learn about competitive internal products. This knowledge may then be used by direct competitors who hire the same outsourcing vendor.

Creating applicable contracts can be complex due to numerous unknowns at a project's initiation. Rao (2009: 76) discusses how this adds overhead in terms of project management time. The timing of specialised resources on a project is also challenging for outsourcing vendors and customers alike. Rao (2009: 76) explains that pre-requisite development overruns cause project scheduling challenges as expensive specialised resources may be unused for long periods of time. These resources may be committed to the vendor's other clients when they do eventually become necessary and therefore unavailable to the original client. This can create bitterness distrust between the vendor and client.

Brooks (2006: 48) advises that outsourcing exacerbates the stresses of IT workers and teams. IT workers may feel that their job security, career development and value to the company are being diminished by outsourced resources resulting in lower productivity, less motivation and higher staff turnover. IT teams often struggle with communicating with outsourced resources, possibly as a symptom of distrust. These issues are often compounded when outsourcing is done offshore due to cultural differences.

3.3.3 Addressing maintenance challenges

There is no easy way to deal with the challenges mentioned. One feels that a great deal of change needs to occur to address all the issues. Software evolution requires a great deal of change as it is a different approach to doing maintenance altogether. Software evolution revolves around the idea that software does not change, but evolves. In this sense, development of a software program never ceases. Software evolution is discussed in Section 3.3.3.1.

There has been a lot of growth in the software development industry in the last 3 decades leading to more intuitive and more functional software development and maintenance tools. Many of these tools can assist in easing the burden of the maintenance developer. This is discussed in Section 3.3.3.2.

3.3.3.1 Software evolution

Mens *et.al.* (2005: 1) put forward that addressing the challenges of maintenance can only be done by making change the core of the software development process. Software maintenance should not be seen as a phase in the process but rather an ongoing process which begins immediately after development. This type of maintenance is most applicable to Lehman's classification of E-type systems. E-type systems are defined as software which is used in the real world and requires constant change due to changing requirements and user demands. As this real world changes, software is required to evolve to accommodate the changing environment (Lehman & Ramil, 2001: 20; Pfleeger, 1998: 87). E-type systems are contrasted to S-type systems which are built to operate in a static-environment and do not require change. Lehman (1980: 1062) had initially also defined a P-type category as an intermediate between the E and S types however Lehman and Ramil (2001: 20) have since stated this category can be included into either of the S or E type categories.

The most common software systems including those which business and organisations rely on for operational purposes are usually of type E. Lehman has proposed a set of "laws of software evolution" (Lehman, 1980: 1068; Lehman & Ramil, 2001: 18) which provide insight into the software process and the nature of software evolution. Pfleeger (1998: 90) recommends that we make use of this insight to design systems to be more flexible and to control the maintenance of software rather than treating software maintenance as a reactive process. These laws are listed below:

" 1. An E-type system must be continually adapted else it becomes progressively

less satisfactory in use.

2. As an E-type system is evolved its complexity increases unless work is done to maintain or reduce it.
3. Global E-type system evolution processes are self-regulating with statistically determinable trends and invariances.
4. Average activity rate in an E-type process tends to remain constant over system lifetime or segments of that lifetime.
5. In general, the average incremental growth (growth rate trend) of E-type systems tends to decline ” (Lehman, 1980: 1068; Lehman & Ramil, 2001: 18).

The first two laws are understandable and self-explanatory. Pfleeger (1998: 90) explains that the third law deals with the predictability of software failures and avoiding unexpected maintenance. Lehman and Ramil (2001: 27) propose system or software project characteristics which influence maintenance requirements be identified. Examples of these characteristics are: size, age, application area, team size or organisational experience. Over time, data should be collected, baselines established and patterns analysed so that a software evolutionary strategy can be developed to avoid unexpected changes.

The fourth law, as simplified by Pfleeger (1998: 90) states that productivity on a software project can reach an optimal level. Adding more resources after productivity has reached this level will not yield higher productivity. Similarly, Lehman and Ramil (2001: 29) explain that one can only improve a software system to an optimal point. Thereafter, performing more maintenance may only keep the system running at that level but would not improve its productivity.

The fifth law states that as a system ages, its growth rate, in the form of planned releases slows down and so too, will its usefulness. Pfleeger (1998: 90) explains that this law implies that new functionality should be added by building a replacement system.

Lehman has formalised a system to assist the software industry in planning for maintenance activities, and not just reacting to them. Software evolution not only involves understanding that software ages, but attempting to pre-empt its aging by understanding of growth degradation patterns. Finally, Lehman discloses that maintenance of software cannot immortalise a software system and one does also need to plan for software replacement.

3.3.3.2 Software maintenance tools

Dishaw and Strong (1998: 152) report that software tools for maintenance can improve maintenance productivity and quality. The findings of Iivari (1996: 102) found this to be especially true when tool usage and the tool adoption process is supported by management. Management adoption may stem from usage of a process adoption tool aimed at software process improvement as many companies have dealt with software process improvement (García *et al., 2*010: 1).

Software process improvement tools are usually modelled to a software process model. Garcia *et al.* (2010: 2) summarise the activities and functionality included in these tools as: collecting and managing general company info, assessing project and process goals, providing questionnaires and self-assessment tasks to software practitioners.

One challenge with maintaining large, decentralised systems is duplication of code snippets or updating of system components used in multiple places. Duplication of code snippets are referred to as code clones (Kim *et al., 2*005: 187). Veerman (2007: 353) relates that the traditional, manual way of individually updating these duplicated items can become very impractical and time consuming as systems grow and evolve. There are many different kinds of tools, e.g. debuggers, test case generators, version control tools and configuration management tools which assist in maintaining systems with multiple duplicated items.

Sulaiman *et al.* (2002: 566) have included tool usage as part of their investigation of documentation. They found that the most common reason for not maintaining documentation is time constraints. Documentation tools do assist in this regard

however these tools do have some shortcomings. Computer-aided software engineering (CASE) tools available do assist in auto-documentation and reverse-engineering programs into design models however this usually only works well if the tool was used from the analysis and design stage.

Software tools have developed a great deal and there are many ways that these tools can assist in addressing maintenance challenges however Cordy (2003: 3) has found that there are practical barriers to adopting these tools:

- Business risk. The cost of developing self-maintaining software is small compared to the cost of a potential error made in maintenance because, as Cordy (2003: 4) explains the system is at risk of becoming non-functional with every small change.
- Technological challenge. Cordy (2003: 5) discusses that the numerous variances in syntax, vendor implementations and official releases of programming languages and software development tools pose a problem to completing software maintenance automation as failures would occur with every variant that is not catered for.
- Social challenge. Maintenance developers might feel inclined to provide negative feedback on maintenance automation tools because, as Cordy (2003: 6) speculates, they might feel as if they could be replaced by the tool or because they don't understand how the tool works.
- Economic challenge. If a software component is currently functioning, it is difficult to motivate to stakeholders to invest in performing preventative maintenance on this component. Cordy (2003: 8) reasons that budgets are generally planned one year in advance and re-architecting tools for a system is cost that will only reap financial returns many years down the line.

3.4 Conclusion

Even though software development has evolved over a fifty year period, the following steps are common amongst most software development methodology processes:

" 1. Inception – Software product is conceived and defined
2. Planning – Initial schedule, resources and cost are determined

3. Requirements analysis – Specify what the application must do; answers “what?”
4. Design – Specify the parts and how they fit; answers “how?”
5. Implementation – write the code
6. Testing – Execute the application with input test data
7. Maintenance – Repair defects and add capability” (Braude & Bernstein, 2011: 33).

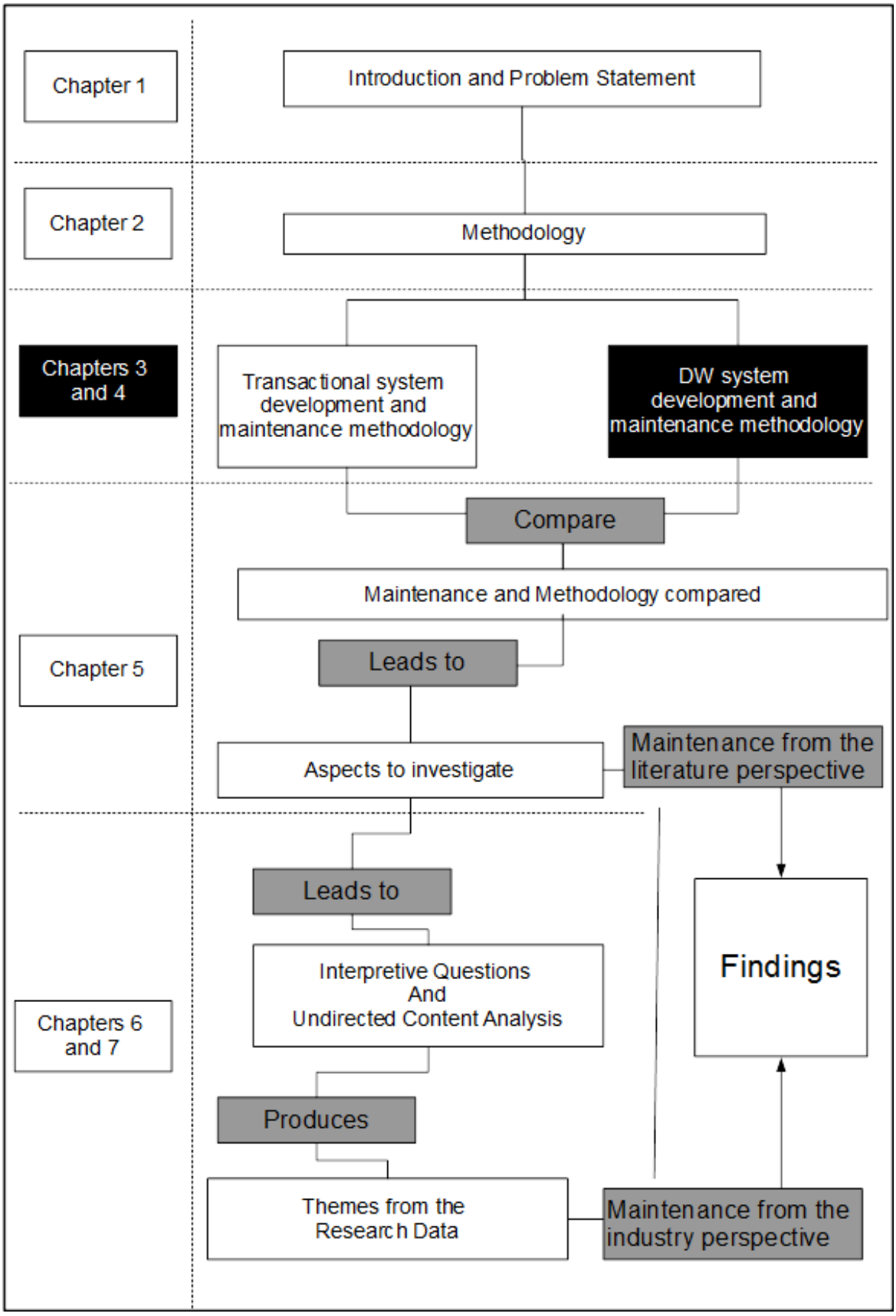
The noticeable difference in the methodologies discussed is a trend toward smaller, iterative development cycles as well as a greater emphasis on user communication.

Software maintenance is usually corrective where bugs are being fixed or adaptive where user requests for change and additions are being addressed. Preventative or perfective maintenance is less common as the costs for this type of maintenance is hard to justify to stakeholders.

Although methodologies, frameworks and standards are in place to perform maintenance, these are rarely used as maintenance is often done in an emergency situation where time saving is essential. In addition to this, software maintainers face many other challenges such as poorly documented systems, maintaining large highly integrated systems, dealing with legacy systems and third party COTS systems.

There aren’t any easy ways to address these maintenance challenges. Lehman has pioneered work in software evolution which tries to minimise these challenges by arranging for maintenance tasks to be ongoing and not just a phase in a software development methodology. Other than changing ones perspective on maintenance, investment in software maintenance tools can assist with addressing maintenance

challenges. Many tools have been developed which assist with documentation, code-clone updates and even the maintenance process. Fully automated maintenance is not impossible; however there are social and economic barriers to overcome before addressing the technological barrier that this poses.


Chapter 1
Introduction and Problem Statement
Chapter 2
Methodology
Chapters 3 and 4
Transactional system development and maintenance methodology
DW system development and maintenance methodology
Compare
Maintenance and Methodology compared
Chapter 5
Leads to
Aspects to investigate
Maintenance from the literature perspective
Leads to
Findings
Interpretive Questions And Undirected Content Analysis
Chapters 6 and 7
Produces
Themes from the Research Data
Maintenance from the industry perspective

Chapter 4
Data Warehouse Development and Maintenance

4.1 Introduction

As this dissertation aims to compare challenges in maintenance between transactional systems and data warehouse systems, it is necessary to examine data warehouse development and maintenance methodologies. That is the purpose of this chapter which is depicted by the blocks shaded in back in the organisational diagram on the previous page. It is necessary to fulfil this purpose of a literature review on DW development and maintenance methodologies as the reader will later require this information to appreciate the discussion on the responses to the interpretive research questionnaire performed in Chapter 6 and the analysis thereof performed in Chapter 7.

“Data warehousing” is a term that is often mixed in with other terms such as “decision support”, “business intelligence” and “management information systems”. In attempt to be clear on meanings, a brief explanation is prudent.

“Decision support” and “management information systems” are older terms and are relatively old fields in computing. Sprague (1980: 4) explains that during the 1960s, management information systems (MIS) had an information focus for report generation and was aimed at middle managers. This evolved into decision support systems aimed at top managers and decision makers. Sprague (1980: 1), Shim *et al.* (2002: 111) and Power (2003: 1) report that the term “decision support systems” (DSS) came into being in the late 1960s or early 1970s and this term described computer technology solutions which can be used to assist with decision making. Shim *et al.* (2002: 114) refers to the data warehouse (DW) as a powerful tool used for building DSS which emerged in the 1990s. Power (2003: 5) relates that Ralph Kimball and Bill Inmon promoted and pioneered the use of relational database technologies in the form of data warehouses for building DSS.

Business intelligence (BI) is a newer term. Kimball *et al.* (2008: 10) conveys that term BI emerged in the 1990s and initially referred to reporting and analysis of data from the DW but is now often used as the term to refer to the entire end to end decision and management support system. Kimball *et al.* (2008: 10) have now combined the terms so that "data warehouse/business intelligence (DW/BI)" refer to the complete system.

In this dissertation, we are concerned with data warehousing. As stated in the above paragraph, the two pioneering authors in this field are Ralph Kimball and William Inmon. These authors have continued to dominate the literature on this topic. Their respective definitions of a data warehouse are given here. Kimball *et al.* (1998: 19) define the data warehouse as "the queryable source of data in the enterprise". Inmon (2005: 31) defines a data warehouse as "a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions. The data warehouse contains granular corporate data."

Aside from decision support, data warehousing and business intelligence, there also exists some ambiguity between the terms "data warehousing" "and online analytical processing" (OLAP) and how these terms relate to online transactional processing (OLTP). These terms are discussed in Section 4.2.

After clarification of industry terminology is addressed, Section 4.3.1 provides an examination of DW 2.0 which is the latest data warehousing framework published by Inmon. Similarly, Section 4.3.2 provides an examination of Kimball's latest edition of what is now being called the "Kimball life cycle". Each of these sections are covered rather extensively and hence they each have subsections for an introduction, discussion of the methodology and its parts, and a summary.

Challenges of data warehouse maintenance is discussed in Section 4.4 and Section 4.5 concludes the chapter.

4.2 OLTP, data warehouses and OLAP

In this dissertation, there is a distinction between data warehousing systems and transactional systems. As is evident from the definitions above, data warehouse systems are designed for querying and analysing data to make decisions efficiently. This is opposed to transactional systems which are designed to update, insert and delete data to run business processes efficiently. In the software industry, distinctions exist between online-transactional processing (OLTP), data warehouses and online-analytical processing (OLAP) systems.

Transactional systems as referred to in this dissertation are the same as OLTP systems and have the same design goals. Microsoft Technet (2000) argues that OLTP systems are not suited to data analysis because: there are difficulties in understanding the relationships among tables, OLTP systems are often divided between multiple servers and running *ad hoc* queries on an OLTP system jeopardises the reliability of the system.

To address these shortcomings of OLTP systems, data warehouses are built to serve as a queryable collection of data in an organisation. Microsoft Technet (2000) states that data warehouses consolidate data structures and data compatibility among multiple OLTP databases, organise data to facilitate analytical queries and allow better performance on query retrieval without hindering the operation of a production system.

Microsoft Technet (2000) does state that the terms data warehousing and OLAP are sometimes used interchangeably however they point out that while data warehouses store the data obtained from multiple sources, OLAP technology is used by client applications to analyse the data. OLAP generally pre-aggregates frequently queried data, enabling faster response. OLAP tools enable users to create views for data which allow for easier analysis of data through quick, interactive summarising; detailing and pivoting. OLAP databases can be stored in a multi-dimensional online-analytical processing (MOLAP) storage engine, a relational online-analytical

processing (ROLAP) storage engine or in a hybrid online-analytical processing (HOLAP) storage engine. Chaudhuri *et al.* (2011: 91) explain that MOLAP storage engines pre-compute and store large amounts of data and indexes based on statistical records of data access. MOLAP requires large amounts of disk space and takes longer time to load. ROLAP data is stored at the databases level in data models optimized for query efficiency and does not require the extra space needed for MOLAP. HOLAP storage uses a combination of MOLAP and ROLAP. HOLAP storage engines generally require a developer to choose portions of data requiring faster access to store using a MOLAP engine and less frequently accessed data is stored using a ROLAP engine.

4.3 Data warehouse methodologies

For the rest of this chapter, the focus remains on the work of Inmon and Kimball and their recommendations on the development and maintenance of data warehouses. Part of the motivation for focussing on these two authors is that they have been acknowledged as pioneers of the field (Power, 2003: 5). Sen and Sinha (2005: 79) have done a comparison of data warehouse methodologies developed by some of the industry's main core-technology vendors, infrastructure vendors, and information modelling companies. Before detailing their comparison of these methodologies, Sen and Senha (2005: 80) state that these methodologies were developed from the definitions and concepts authored by Kimball and Inmon. Thereafter, Sen and Sinha (2005: 81) tabulate the methodologies being compared and detail whether these methodologies follow Inmon's enterprise data warehouse design philosophy, Kimball's dimensional modelling design philosophy or both. It is thus evident that the industry uses key design philosophies from Kimball and Inmon.

Technology is constantly changing and methodologies evolve to cater for these changes hence the latest editions of the work of these authors are utilised as the preferred reference for this chapter.

4.3.1 Principles of Inmon's DW 2.0

The discussion in this section is centred on Inmon *et al.'s* (2008) DW 2.0 book. At the time of writing, this is the latest edition to his books related to data warehouse development methodology.

In DW 2.0, the term "corporate information factory" is sometimes used to refer to the entire system and we shall assume that the terms DW 2.0 and corporate information factory are synonymous In a previous publication Inmon *et al.* (2001:11) define the corporate information factory as: "the corporate information factory (CIF) is an architecture—an infrastructure—for the information ecosystem, consisting of the following components: external world, applications, integration and transformation layer (I & T layer), operational data store (ODS), data warehouse, data mart(s), internet and intranet, metadata repository, exploration and data mining data warehouse, alternative storage, decision support systems (DSS)" (Inmon, Imhoff, & Sousa, 2001: 11)

4.3.1.1 Introduction

Inmon *et al.* (2008: 23) state that "DW 2.0 is the definition of data warehouse architecture for the next generation of data warehousing". The following four traits of DW 2.0 are summarised by Inmon *et al.* (2008: 54): the recognition of life-cycle data during its passage within the data warehouse, the inclusion of unstructured data in the data warehouse, the inclusion of metadata and; the ability of adapting to change.

The passage of life cycle data into the warehouse is the first trait in DW 2.0. life cycle data has four major sectors

"
- Interactive sector, where data warehousing is done in an update mode at transaction response-time rates;
- Integrated sector, where data is integrated and analytical processing can be done;

- Near line sector, which serves as a caching area for data in the integrated sector; and
- Archival sector, where data that may still be needed goes after a significant decline in the probability that it will be accessed" (Inmon *et al., 2*008: 54).

The second trait is the inclusion of structured data. Structured data is data that is formatted in a repetitive, predictable way. DW 2.0 is meant to be designed around both structured and unstructured data sources. Dayal *et al.* (2009: 10) report that unstructured data could account for more than 80% of an enterprises information as these include all text documents, images, video, audio and other different forms of spatial and temporal data.

The third trait of DW 2.0. is that metadata is defined simply as data about data (Lunan Li, 2010: 125; QianCheng, 2010: 340). The inclusion thereof means storing and keeping track of data keys, indexes, data types, data constraints, age of data etc. Inmon *et al.* (2008: 33) explain that metadata allows the user to view previous analysis done on data or create different types of relationships between data. Systems built using metadata also streamlines the end user experience as well as administrative tasks. Due to the growing size of data warehouses, the use of metadata is a compulsory in any DW system.

The final trait is ability to adapt to change. Changing requirements have always been a thorn in the side of a software project development team. Bhatti *et al.* (2010: 319) explain that changing requirements could be due to external factors e.g. customer's pressure, changing business needs or internal factors e.g. poor understanding, technical unfeasibility of customer's requirements. Harker *et al.* (1993: 266) reason that there must be a response to these changing requirements especially requirement changes due to better understanding as development continues. Inmon *et al.* (2008: 112) emphasise the difficulty in making changes to underlying infrastructure as this is often procured at the early stages of a project where vendors work according to a specification which managers have budgeted for. Making changes to infrastructure requires the involvement and agreement of multiple parties.

Inmon *et al.* (2008: 115) stress that the effect of changing requirements on infrastructure needs to be addressed quickly and early when the requirement becomes known. The effect of acting on a requirements change can be mitigated by the separation of temporal and stable data. Temporal data changes frequently e.g. bank account balance whereas stable or static data rarely changes e.g. bank account number. As stable data does not change, it should be stored separately so that when a change is required to infrastructure, it only affects the temporal data. Effectively, the volume of data affected by the change become smaller, therefore it becomes easier to take a snapshot of this data before making the change thereby mitigating risk caused by changing infrastructure. The separation of temporal and stable data therefore allows quicker response times to changing infrastructure requirements.

Another way to deal with the problem of changing requirements is by embracing the change. Inmon *et al.* (2008: 124) advocate changing business requirements for a data warehouse system after it has been developed because it infers that business is using the data warehouse and find value in adding more to it. Changing business requirements in traditional transactional systems does not enjoy the same notions. Change in business requirements is catered for by the methodology employed by developing a DW 2.0 type system.

Although there is mention of gathering requirements from users during the data model, there is no definitive requirements gathering phase. The DW 2.0 methodology builds on the spiral methodology discussed in Chapter 3. DW 2.0 does not have definitive phases but rather defines program streams. These streams are discussed in Section 4.3.1.2 and Section 4.3.1.3 examines the data model in DW 2.0.

Further to the lack of definitive phases, is the lack of a definitive maintenance phase. As discussed in Chapter 3, maintenance may be reactive e.g. corrective

maintenance or pro-active e.g. preventive maintenance. Section 4.3.1.4 examines the maintenance in DW 2.0.

4.3.1.2 Methodology

Sin and Senha (2005: 80) point out that “Data warehousing methodologies share a common set of tasks, including business requirements analysis, data design, architecture design, implementation, and deployment” However, DW 2.0. does not follow these tasks in the traditional way. There are no defined, sequential development phases as one normally expects to find when researching software development.

The approach used in DW 2.0. development is a spiral development approach as described in Section 3.2.3. Inmon *et al.* (2008: 124) reason that a cyclical development approach is required because the customer can make use of the data warehouse, understand what the data warehouse is capable of delivering and then modify their requests and/or make new requests based on their newly gained understanding. This allows the initial cycles of development to be in the form of working prototypes. Other reasons for choosing a spiral methodology include: the allowance of overlapping tasks, management of expectations or risk mitigation and quick delivery cycles.

Due to the lack of definitive phases, it may be unclear where to begin a data warehouse project. Software development generally begins with feasibility analysis and user requirements. Boehm (1988: 66) defines that the spiral methodology begins with a feasibility analysis and an understanding of current operations. Thereafter a top-level requirements specification is performed. Inmon has previously stated that “the requirements for the building of the data warehouse come from a data model. End-user requirements are a part of the picture, although the end-user requirements enter the picture indirectly” (Inmon, 2005: 377). This has changed in DW 2.0 as Inmon *et al.* (2008: 166) has designated user input as the vital source for design of the data model which is built at the beginning of a DW 2.0 system.

It is stressed by Inmon *et al.* (2008: 128) that the spiral methodology should not be used to create multiple isolated data marts but rather to continuously create reusable programs and structures where success is measured in the long term. To achieve this, Inmon *et al.* (2008: 129) has developed a framework, called "the seven streams approach" to build enterprise data warehouses using the spiral methodology. In this framework, each stream run simultaneously, in parallel as there is minimal dependency between streams and a stream does not require completion before the next can begin. Hence, these are not to be regarded as development phases, but rather as development streams. The following is a summary of the seven streams in Inmon *et al.'s* seven streams approach:

1. Enterprise reference model stream – a corporate data model, or high level logical view of the corporation is built and continuously maintained. This is done by breaking down the corporation into subject areas and iteratively building each subject area's model (Inmon *et al., 2*008: 130).
2. Enterprise knowledge coordination stream – the data and models from various business modelling exercises and data collection exercises needs to be coordinated to avoid redundancy and enhance understanding. Besides the corporate data model, Inmon *et al.* (2008: 131) also stipulate the development of an "information factory" and undergoing the exercise of data profiling. The information factory development stream and data profiling stream are separate streams discussed next. It is vital that the knowledge gained in these streams be coordinated iteratively in this stream. The output of this coordination is a steady-state model which aims to provide reliable information about the corporation's data.
3. Information factory development stream - Information factory development entails identifying data within existing systems which allow business to analyse key concerns e.g. growth and profit. As these questions usually require all data to be analysed to give a complete answer, development takes place one subject area at time, iteratively growing toward a complete answer. The subject areas which are tackled first are based on business priority hence the subject areas in development of the information factory drives the priorities of the other streams (Inmon *et al., 2*008: 133).

4. Data profiling and mapping stream - Data profiling involves gaining understanding of current system data; where this data may be duplicated and where incorrect or incomplete system data occurs. Mapping the best source areas to target structures in the data warehouse follows.
5. Data correction stream – this involves going through source system data which links to relevant data warehouse data and correcting it, adding to it or deleting necessary records from it.
6. Infrastructure stream – "This stream of activities addresses the supporting infrastructure for the scalability of the information factory, including consideration of people, resources, platforms, tools, policies, standards, and procedures ... The infrastructure stream is undertaken component by component" (Inmon *et al., 2*008: 133).
7. Total information quality management stream (TIQM) - Inmon *et al.* (2008: 134) recommend the TIQM method developed by English (2004). This is an ongoing process where sample data is continuously extracted, measured quality assured by reporting on the measurements. TIQM is a habit that develops within an organisation where good data quality impact is measured and data quality improvements are standardised.

Inmon *et al.* (2008: 129) state "The basic premise is that business intelligence initiatives should be seen as programs, not projects." Hence one does not find clear stages; *viz.* requirements gathering, programming, implementation, maintenance as one would find in a project, typically in done using the waterfall methodology which Inmon *et al.* (2008: 12) refers to as the exact opposite of the spiral methodology. The seven streams approach to building a data warehouse revolves around the system's data.

4.3.1.3 Data design

The data model in DW 2.0 brings together the different people, programs and processes in DW 2.0. It is a design of the different data components of business (Inmon *et al., 2*008: 158). The scope of the data model is to be decided upon. Inmon *et al.* (2008: 158) refer to the scope of integration as a statement of what is

included in the data model and the level of detail of the inclusions. The level of detail is also referred to as the granularity.

At a high level, Inmon *et al.* (2008: 159) model data as an entity relationship diagram (ERD) which shows the relationships between various subject areas of the business. At mid-level, data is modelled per data item set, which shows actual table names, unique identifiers and column names. At the low level, the data model includes foreign key relationships, indexes and other data constraints.

Data is modelled around the life cycle data sectors. These are summarised in Table 4-1 by the researcher. In the interactive sector, a data model is built for each application which results in smaller, performance driven data models. In the interactive sector, Inmon *et al.* (2008: 56) explain that warehouse activities and data volumes are small for quick access. Data in this sector can be real-time and applications may access this data directly without data passing through an extract transform load (ETL) layer. Historical analysis does not occur in the interactive sector hence Inmon *et al.* (2008: 60) stipulate that little historical data is stored in this sector and the level of granularity between the various applications are not consistent.

Before data is moved to the interactive sector for analysis, Inmon *et al.* (2008: 163) highlight that a data model needs to be developed which converts, transforms and integrates the data for the integrated sector. This imperative data model's role is to merge all data from different applications in the interactive sector. This is called the corporate data model. This layer is depicted in Table 4-1 as an intermediate layer without table borders. Inmon *et al.* (2008: 62) stress that "many conversions that have to be done to transform operational application and interactive transaction data into corporate data". Inmon (2005: 79) explains that the corporate data model is the beginning point for other existing data models.

INTERACTIVE SECTOR		INTEGRATED SECTOR	NEAR LINE SECTOR	ARCHIVE SECTOR
• Data model created for each application • Low volumes of data • Fast access • Can be real-time • ETL not necessary • No historical analysis • Stored on disk	*The corporate data model* • Middle ground between interactive sector and integrated sector • Conversions and transformations are mapped • Merges data from various applications in the interactive sector. • Starting point for other data models.	• Multiple types of data structures. • Differing levels of granularity. • High demand of data for query and analysis. • Data cannot be updated or deleted. • Updates are always handled by inserting new records.	• Optional sector used to save cost and increase performance of integrated sector. • Exact copy of data models of integrated sector hence no transformations. • Contains less frequently access data. • Storage on cheaper robotic cartridges. • Data is sometimes moved back to integrated sector.	• Contains data that is not useful in the foreseeable future. • Decision must be taken regarding data model and data format. • Data access could take days.

Table 4 - 1 Summary of data sectors in DW 2.0

Inmon (2008: 62) explains that many types of structures and multiple levels of data granularity exist in the integrated sector as multiple types of analysis occur in this sector. Detailed, subject oriented data, stored at a granular level is used for small queries where results change frequently. Summary data which does not change frequently is stored for quick access to these queries e.g. annual totals. Inmon *et al.* (2008: 63) describe continuous time span data as infrequently changing data e.g. customer address. This data is stored and rarely updated. Linked to the subject oriented data is profile data. This is collected from multiple sources in the interactive sector to provide detailed information about the subject in a single record e.g. a customer's demographics, location, earning and spending potential. Data in the integrated sector is updated through an ETL process. As the data in this sector is in high demand and there are various types of querying going on, Inmon *et al.* (2008: 66) suggest a schedule of allocated times for different types of queries in order to enhance the query response times. When data in the integrated sector gets out-dated, new records are created based and linked to the time dependent record. Inmon *et al.* (2008: 69) stipulate that no deletions or updates occur in the interactive sector unless data is being moved to the near line sector or the archival sector.

When data is no longer useful for querying or analysis, it is moved from the near line sector or integrated sector to the archive sector. Inmon *et al.* (2008: 76) explain that a company may be legally required to keep data for certain periods of time. Alternatively, data is archived because of the possibility that it may be useful in the future. Archive data contains decades of company data hence data retrieval could take long periods of time to retrieve as the disconnected physical media may be stored offsite. Inmon *et al.* (2008: 84) enlighten that a choice should be taken regarding the data model of the archival sector. If it is expected that the data will

be read back in to the integrated sector, using the same database tools as the present, then using the same data model as the integrated sector would be advisable. However, if it is expected that the toolset may change, it is advisable to transform the data into more standardised and accessible format e.g. a flat file format.

Inmon *et al.* (2008: 166) suggest the inclusion of unstructured data when considering data modelling. Although unstructured data does not lend itself to modelling, a classification system of unstructured data may be modelled and developed.

4.3.1.4 Maintenance

As mentioned, DW 2.0 does not contain definitive phases hence there is no defined maintenance phase. However, combination of the spiral methodology and the seven streams approach does cater for data warehouse maintenance during development. In Chapter 3, maintenance was broken down into corrective, adaptive, preventive, perfective and emergency maintenance. Data warehouses are not business processing systems therefore a discussion on emergency maintenance is not applicable. Corrective and adaptive maintenance have much commonality; as do preventive and perfective maintenance. The discussion on maintenance is therefore divided into two parts: corrective and adaptive maintenance is discussed first and then preventive and perfective maintenance is discussed.

Corrective and adaptive maintenance

In DW 2.0, corrective and adaptive maintenance is catered for by the cyclical nature of the spiral methodology. Inmon (2007) passionately explains that it is in the nature of iterative development for users find issues with the data warehouse system and provide feedback on each release cycle. Action is taken on the next iteration or through a shortened change process as described in the paragraphs below.

Adaptive maintenance is triggered by business requests for additions and changes. Inmon (2008: 111) emphasises that DW 2.0 needs to be prepared for change requests from business as business is constantly changing and "business requirements change faster than the rate at which IT can respond to change" (Inmon *et al., 2*008: 114). Some companies react to this problem by freezing business requests so that IT can catch up to the requests already made or by spending more money and hiring more staff to deal with the business requests. Inmon *et al.* (2008: 115) advise that neither of these options is viable and that IT needs to shorten response times on business requests.

DW 2.0 goes some way in assisting to shorten business requests by prescribing the physical separation of changing or dynamic data and data that remains unchanged or static over long periods. By dividing dynamic and static data onto different physical storage locations, Inmon *et al.* (2008: 118) explain that the data which remains static does not need to go through large and data migration and manipulation tasks whenever there is a request from business which requires data restructuring. Only the dynamic which is constantly changing needs to undergo these difficult, time consuming operations hence the change request process turnaround time is reduced.

Inmon *et al.* (2008: 121) also suggest taking snapshots of data every time a business request is made. This way, the request should occur to data going forward only. When querying the data, Inmon *et al.* (2008: 121) concede that the query needs to be delimited by time. However, one would imagine that a query would need to traverse through a list of rules if spanning through large time frames. Nonetheless, if this rules list is updated every time a change is made, it could make the effects of change to data structures less significant.

Another benefit of creating a rules list for each change is that it allows a historical record of rules to be created and referenced from the data warehouse. Inmon *et al.* (2008: 121) explain that it becomes easier for the analyst to recognise the difference in calculations based on the rules list of changes as they happened.

As part of the DW 2.0 methodology, Inmon *et al.* (2008:96) emphasise that metadata is incorporated into each sector of the data warehouse. This is important in corrective and adaptive maintenance as metadata stores information such data sources, descriptions and the type of analysis that has already been done. These are the reasons why Inmon *et al.* (2008: 95) underline that metadata is important to the maintenance developer.

Preventive and perfective maintenance

Much of the maintenance performed in DW 2.0 is of a preventive or perfective nature. This is evident in the seven streams framework as preventive and perfective maintenance is dealt with in the enterprise knowledge coordination stream and the total information quality management stream (TIQM).

Monitoring the DW 2.0 is part of TIQM. Inmon *et al.* (2008: 168) suggest the following three types of monitoring:

1. Transaction monitoring: Response times of transaction in the interactive sector are measured for consistency. Application's use of resources and data indicate where improvements could be made. Monitoring peak period processing over time gives a good indicator of system growth and allows for system capacity planning.

2. Data quality monitoring: Data that is passed through an ETL process into the integrated sector should be checked for data format correctness, data outliers and foreign key correctness. Incorrect data should be dealt with in an automated fashion e.g. flagging the data as incorrect so that the user is aware of any inconsistency.

3. Data warehouse monitoring: Usage frequency of data is monitored to assess whether data should be moved to the near-line or archival sector. This type of monitoring is done via the structured query language (SQL) code that accesses the data. The purpose of this monitor to reduce the amount of data in the integrated sector, thereby reducing cost and access times.

Monitoring generally is successful at finding data problems. The following are suggested strategies for dealing with these problems:

"
- Fix the source data: Actually go into the data store and physically repair the data.
- Fix the source program: Apply the correct edits to validate the data.
- Adjust for changes: Recognize and resolve situations in which data attributes are being used for a purpose other than their original intent, e.g., a gender code that has more than two distinct values.
- Transform the data: Transform the data on the way into the data warehouse—the most common strategy, but not the only strategy that should be employed" (Inmon *et al., 2*008: 286).

Whilst the suggested strategies above are generally considered corrective, it is important to note that Inmon *et al.* (2008: 285) suggest that these data issues should be searched for through on-going data profiling. Therefore, data issues should be prevented from being found by a business user.

The data flow strategy of DW 2.0 ensures that the frequently accessed interactive sector of the system does not contain unnecessary data thereby assisting in high speed of data access. The data flow strategy includes the implementation of the near line sector and archival sector where infrequently accessed data is moved to. Inmon *et al.* (2008: 161) explain that performance considerations are incorporated into the data model. In other methodologies, these types of considerations are usually part of a perfective maintenance strategy.

4.3.1.5 DW 2.0 summary

The architecture and data design is centred on the four data sectors viz. interactive sector, integrated sector, near-line sector and archival sector. The corporate data model is an intermediate layer which contains the design of the ETL processes between the interactive and integrated layer.

The development methodology employed by DW 2.0 is the cyclical spiral methodology. This methodology is used in conjunction with the seven streams

approach developed by Inmon. The seven streams approach is a framework which identifies seven areas of development for a data warehouse system. These areas are dependent on each other and development may overlap on some areas. No single area is developed in isolation and each area is continuously worked on through iterative development.

Maintenance in DW 2.0 is performed both in an adaptive and preventive fashion. A key logical distinction is made between data which is frequently changing (or temporal data) and data which remains static. Making this distinction allows the physical separation of this data, resulting in changes to temporal data taking less time to perform and occurring with less of a ripple effect to other parts of the system. It is stressed that a good data warehouse is one which continuously receives new and changing business requests as this is evidence of the usage and value of the data warehouse.

4.3.2 The Kimball life cycle approach

The discussion in this section is centred on Kimball *et al.'s* (2008) "Data Warehouse Life Cycle Toolkit". At the time of writing, this is the latest edition to this author's books related to data warehouse development methodology.

4.3.2.1 Introduction

Kimball *et al.* (2008: 3) explain that the term "Kimball life cycle approach" came to being as this was the popular term used for what the authors originally referred to as the business dimensional life cycle. Due to its popularity, the author also adopted this term.

Kimball *et al.* (2008: 2) list three fundamental concepts for a successful data warehouse:

"
- Focus on the business.
- Dimensionally structure the data that's delivered to the business via *ad hoc*

queries or reports.

- Iteratively develop the overall data warehouse environment in manageable life cycle increments."

Figure 4-1 shows "Kimball Life Cycle diagram" (Kimball *et al., 2*008: 3) which depicts the milestones that need to be achieved for a data warehouse deployment. The above three concepts are evident in this diagram as there is a large block focussing on business requirements as a prerequisite to other milestones. Following business requirements is a dimensional modelling step which occurs parallel to the technical architecture design and BI application design. Finally, after an iteration is deployed, another is prompted by maintenance and growth.

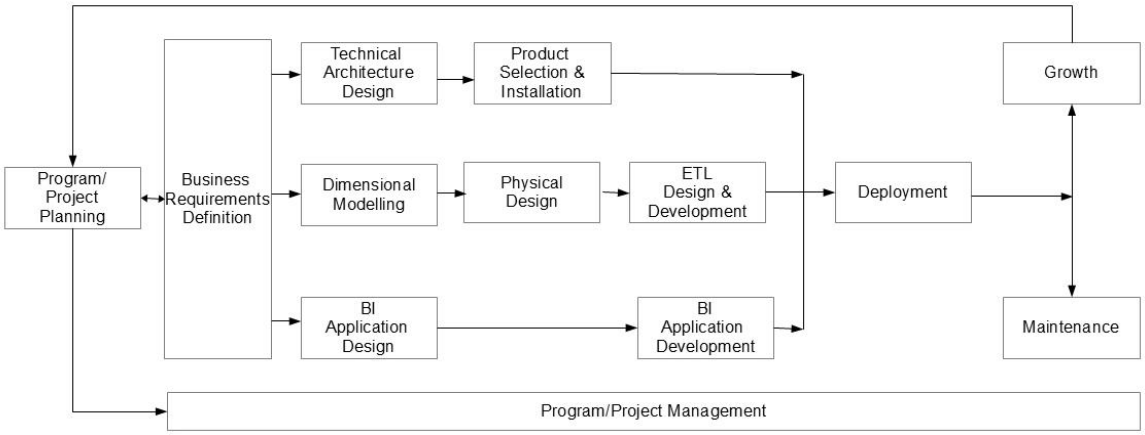


Figure 4 - 1 "The kimball life cycle diagram" (Kimball *et al., 2*008: 3)

The sections that follow contain discussions on different phases of the Kimball life cycle. Program/project planning is discussed in Section 4.3.2.2. Program/project management and the Kimball life cycle requirements definition are discussed in Section 4.3.2.3 discussed in 4.3.2.4 respectively. The phases in the top row of Figure 4-1, namely the technical architecture design and product selection and installation; are referred to as the "technology track" by Kimball *et al.* (2008: 5). The discussion on the technology track is provided in Section 4.3.2.5. The phases in the middle row of Figure 4-1 are known as the "data track" (Kimball *et al., 2*008: 6). The data track includes the dimensional modelling, physical design and ETL design and development phases. These phases are discussed in Section 4.3.2.6. The

“business intelligence application track” (Kimball *et al., 2*008: 7) includes BI application design and development phases of Figure 4-1. This is discussed in Section 4.3.2.7. Deployment is briefly touched on in Section 4.3.2.8 followed by maintenance and growth in Section 4.3.2.9. A short summary of the Kimball life cycle is provided in Section 4.3.2.10.

As this dissertation’s focus is on data warehouse methodology including development and maintenance, the sections that shall be focussed on will be program/project and planning, the data track and the section on maintenance and growth.

4.3.2.2 Project/program planning

In Figure 4-1, Kimball *et al.* (2008: 5), refer to the entire data warehouse as a program and a single iteration is referred to as a project. Kimball *et al.* (2008: 16) begin program/project planning by conducting a data warehouse readiness assessment using the following three factors:

1. Strong senior business management sponsors: The focal factor in assessing readiness is having a respected leader(s) from the business take ownership of the data warehouse program. Kimball *et al.* (2008: 16) refer to this person as the business sponsor and describe him or her as one who is passionate, yet realistic about the potential benefits of a data warehouse. The business sponsor should not be somebody from IT e.g. the chief information officer (CIO).

2. Compelling business motivation: As the Kimball life cycle is driven by business, involves business during as the life cycle progresses and is going to be used by business, the demand for the data warehouse needs to originate from business. DW/BI systems that align with strategic business motivations and initiatives stand a good chance of succeeding. Likewise, business justification becomes a non-issue if the DW/BI system is poised to support a compelling business motivation” (Kimball *et al., 2*008: 17).

3. Feasibility: Kimball *et al.* (2008: 17) consider data availability and quality to be the major concern regarding feasibility of data warehouse development. This type of feasibility can be assessed using data profiling tools.

Once a data warehouse project is deemed feasible, a project scope, charter and plan need to be developed. Kimball *et al.* (2008: 22) advocate that the scope remain focused on only one business process throughout a project iteration. This is especially valid for early iterations of a data warehouse development as one process is more manageable and one business generally extracts data from just one source. The scope definition is done jointly by IT and business. Apart from the scope, the project charter documents: "focus, motivating business requirements, objectives, approach anticipated data and target users, involved parties and stakeholders, success criteria, assumptions and risks." (Kimball *et al., 2*008: 25). It may also be pertinent to include in the charter items that will not be included in the project or project iteration.

To obtain funding for a project, a return on investment (ROI) document should be drawn up. Kimball *et al.* (2008: 27) detail the items that might be included in this. In summary, the costs should include hardware, software, staffing, maintenance and growth. The income should include estimates on tangible items e.g. revenue increases, business cost reduction as well as intangible items e.g. better access to information and customer satisfaction.

The actual planning of the project begins after the scope and charter have been drawn up and the funding has been approved. The project plan begins with a name which the business and IT can jointly create. A project name can be used as a marketing tool to create awareness in business and increase project staff morale and team spirit (Kimball *et al., 1*998: 64).

The project plan includes the staff required for the project. Kimball *et al.* (2008: 33) suggest many roles from both IT and business. Within each role, there may be multiple members who may work only on some iterations of the data warehouse

deployment or the entire program. It is also noted an individual staff member may take the part of multiple role members and that it is rare for there to be a staff member allocated to each role. Table 4-2 lists the roles and members contained in those roles.

Role	**Members**
Front Office: sponsors and drivers	• Business sponsor/business driver • DW/BI director/program manager
Coaches: project managers and leads	• Project manager • Business project lead
Regular line-up: core project team	• Business analyst • Data steward/quality assurance analyst • Data architect/data modeller/database administrator • Metadata manager • ETL Architect/ETL developer • BI architect/BI Application developer/BI portal developer
Special teams	• Technical architect/technical support specialist • Security manager • Lead tester • Data mining/statistical specialist • Educator
Free agents	• Skilled consultants

Table 4 - 2 Kimball life cycle roles and members summarised from Kimball *et al.* (2008: 33)

Once the roles and members have been identified, planning of tasks scheduling, resource allocation, and task and resource dependencies can commence. This is what the overall project plan entails. Kimball *et al.* (2008: 41) provide a generic example of a project plan however emphasise that the company should use software, framework and methodologies which are already in place for planning. It is also emphasised that the project plan needs to keep up to date as resource availability changes and more information becomes available. It is recommended

that project manager get input for each life cycle activity from the major stakeholders for that activity.

A documented communication plan is also recommended by Kimball *et al.* (2008: 44). This plan contains a schedule of which parties with whom communication is to occur e.g. project team, sponsors, business community, etc. The communication plan also details frequency of communication; type of communication e.g. face-to-face, newsletter, etc.; and key topics of the communication e.g. overall progress and funding is discussed with sponsors. A communication plan ensures communication is proactive and prevents unwanted surprises and disappointment.

4.3.2.3 Project/program management

From Figure 4-1, we can see that project management plays a key role through an iteration and program management occurs throughout the development of the data warehouse. Kimball *et al.* (2008: 88) place emphasis on the on-going tasks of project and program management. These include regular project status monitoring and communication through meetings and reports to manage the expectations of the business and sponsors. The project plan and scope need to be updated regularly to ensure so that funding, resource and schedule issues can be tackled early. Kimball *et al.* (2008: 26) differentiate between scope refinement and scope creep as scope refinement does not entail adding to the budget or schedule, but merely ensuring the scope as business requirements become more clearly defined.

On a longer term, data warehouse program development level; Kimball *et al.* (2008: 55) recommend the establishment of governance committees consisting of business leaders and sponsors who meet quarterly or bi-annually to put processes in place including prioritisation, funding, ensuring that IT an business remain aligned. To ensure consistency, data stewardship should transcend individual project iterations. “The data steward is responsible for driving organizational agreement on definitions, business rules, and permissible domain values for the data warehouse data, and then publishing and reinforcing these definitions and rules.” (Kimball *et al., 2*008: 35). Regular feedback from business should be sought out. The program manager

needs to assess the overall usage and customer satisfaction of data warehouse users. The communication plan should to be upheld throughout the data warehouse program development.

4.3.2.4 Requirements Definition

Kimball *et al.* (2008: 33) place a lot of emphasis on requirements collection. This emphasis is depicted in Figure 4-1 as the size of the business requirements definition phase is much larger than any other phase. The reason put forward for this emphasis by Kimball *et al.* (1998: 34) is that a data warehouse designers require better understanding of key business issues than traditional database designers.

In order to attain this level of insight into the business, communication and understanding with business users needs to be fostered. Kimball *et al.* (2008: 65) recommend speaking to business users about objectives of their tasks, the challenges experienced in executing these tasks and the process involved in decision making. This type of information gathering should occur on a face to face level. Kimball *et al.* (2008: 66) prefer one on one interview and facilitated group sessions over questionnaires and surveys.

An understanding of the data content and the ability of IT to deliver business requirements needs to occur alongside the business requirements. Kimball *et al.* (2008: 66) warn that one needs have a realistic idea of the information that the data warehouse can deliver.

Requirements gathering should occur on both a program and project level. Kimball *et al.* (2008: 65) have tabularised the difference in the requirements gathering approach for these two exercises. It also serves as a good indication of the tasks involved in requirements gathering. This table is shown as Table 4-2 below.

	Program Requirements	Project Requirements

Preparation	Broader organisational research.	Focus preparation around single business process.
Participants	Cross functional horizontal representation; more heavily weighted to senior business management.	Potential users/beneficiaries of improved access to single business process (not necessarily single business department); more heavily weighted to middle management and analysts.
Business Questioning	High level understanding of business goals, opportunities, and key supporting processes.	More detailed information and analysis requirements; more inclusion of existing reports and analyses.
Data Audit	Preliminary feasibility assessment of multiple potential data sources.	Preliminary data profiling of single primary data source.
Documentation	Narrative on each process-centric requirement; include preliminary data warehouse bus matrix.	Detailed findings related to three impact areas: data model, BI reports/analytics, and technical architecture.
Next Steps	Prioritization with senior management, loop back to life cycle project planning, and prepare for detailed project requirements.	Business acceptance of findings; updated project scope, if appropriate; move ahead with next Life cycle tasks.

Table 4 - 3 Differences between program and project requirements definition efforts (Kimball *et al., 2*008: 65)

4.3.2.5 Technology Track

The "technology track" (Kimball *et al., 2*008: 5) depicted on Figure 4-1, includes the "technical architecture and design" and "product selection and installation" phases. "The technical architecture is the overall plan for what you want the DW/BI system to be when it's ready for serious use. It describes the flow of data from the source

systems to the decision makers and the transformation and data stores that data goes through along the way. It also specifies the tools, techniques, utilities and platforms needed to make that flow happen." (Kimball *et al., 2*008: 112).

Kimball *et al.* (2008: 110) compare the benefits of a sound technical architecture to that of blueprints for a construction project. A technical architecture assists in keeping to business requirements as the architecture must be designed to address business requirements. As the DW project or overall program progress, the architecture serves as a communication tool for the project as it educates upward management and users of the complexities involved and allows the IT staff to point out what the overall progress is and what is currently being developed. The creation of the architecture plan allows the project manager to ensure that nothing is amiss from the technical aspect in the overall program/project plan. As architecture plan allows one to anticipate issues, and automate the handling of these issues. As such, Kimball *et al.* (2008: 111) explain that it aids preventative maintenance. New team members are brought up to speed faster as the architecture plan demonstrates what is being done and what needs to be done.

The Kimball life cycle architecture is metadata driven as the metadata defines the elements of the architecture and how they intertwine (Kimball *et al., 2*008: 117). A metadata repository assists in maintenance as information about components is accessible and updateable.

There are three major parts to the architecture design for the Kimball life cycle: the back room architecture, the presentation server architecture and the front room architecture. The back room architecture is where the ETL processing occurs. Kimball *et al.* (2008: 120) recommend that an ETL system needs to be able support the production systems, needs to be user-friendly in order to reduce the complexity involved in an enterprise-wide ETL system and an ETL system should be metadata driven so that changes to ETL are easily executed and data flows are documented.

Kimball *et al.* (2008: 133) describe the presentation server layer in the architecture as where portions of the data warehouse is stored and formatted for consumption by different front end applications e.g. BI online analytical processing (OLAP) tools and reporting tools. The presentation layer should fulfil requirements for access to data from all major business processes. Kimball *et al.* (2008: 133) explain that data should be available to all departments from the perspective that a particular department would be interested in e.g. logistics department would require data from distribution centres. The presentation layer should provide access to both summary and detail data. This should include the ability to view details across different department data and to drill into detail data after from summary data. Kimball *et al.* (2008: 134) also emphasise that the presentation layer should serve as a single source for analytic data from which all department draw information consistently as opposed to department with individual data marts debating over which department's information is more correct.

In the data warehouse architecture, the front room is where the user interface and applications are housed. From the user perspective, the front room and its contained applications is the data warehouse because users are often unaware of the complexities of the back room and presentation layer (2008: 141). The applications executed on the front end layer include: direct, *ad hoc* queries initiated by user query applications; standard, scheduled reports; analytic applications often used for budgeting and forecasting; dashboards and scorecards for summarised graphical information; data mining models and some operational queries including real time or near time data. Kimball *et al.* (2008: 143) instruct that the client applications may be Internet or intranet web pages, mobile applications or more in-depth and demanding analytical applications.

Apart from the logical division of the data warehouse system into back room, presentation layer and front room architectures, Kimball *et al.* (2008: 156) also discuss the physical aspects of architecture such as hardware and software choice for servers and for desktops and workstations and the creation of the architecture plan. These topics are part of the Kimball life cycle technology track and should be

considered on implementation; however it is out of the scope of this dissertation to be discussed here.

4.3.2.6 Data Track

The data track includes the dimensional modelling, physical design and ETL design and development phases. Physical design focuses on creation of databases, schemas and other technical aspects which may be dependent on the technology being implemented. Hence, the focus in this section is on dimensional modelling and ETL design.

Dimensional modelling

In the Kimball life cycle, data is structured according to a dimensional model. Kimball *et al.* (1998: 144) describe dimensional modelling to be a logical design technique which uses a restricted type of relational modelling to create a more intuitive and high performance model. Figure 4-2 shows an example of a dimensional model, as depicted in (Kimball *et al., 1*998: 145). The arrangement of tables and joins make it seem like a star shaped structure hence this is often called a star schema. There is one central table, called a fact table and other dimension tables surrounding the fact table and joining to it. Each star join represents one business process, in this case sales. The surrounding dimension tables store data about the attributes of the sales process which help identify individual sales e.g. a sale which occurred at a particular store, during a particular time frame, for a particular product. The central fact table link all the attributes together as they all join to this table. Furthermore, the fact table contains the measurable information about the process e.g. the sales amount and profit.

Kimball *et al.* (1998: 147) strongly advocate the use of dimensional modelling for data warehouse development. A dimensional model is a predictable standard used and understood by developers and software interfaces. The fixed model allows BI applications and database systems to make strong assumptions around usage and key constraints thereby allowing for improved performance. The design further

proves resilient to changes as existing fact and dimension tables can accommodate new facts and attributes without having to reload the entire table and changing the front end BI applications.

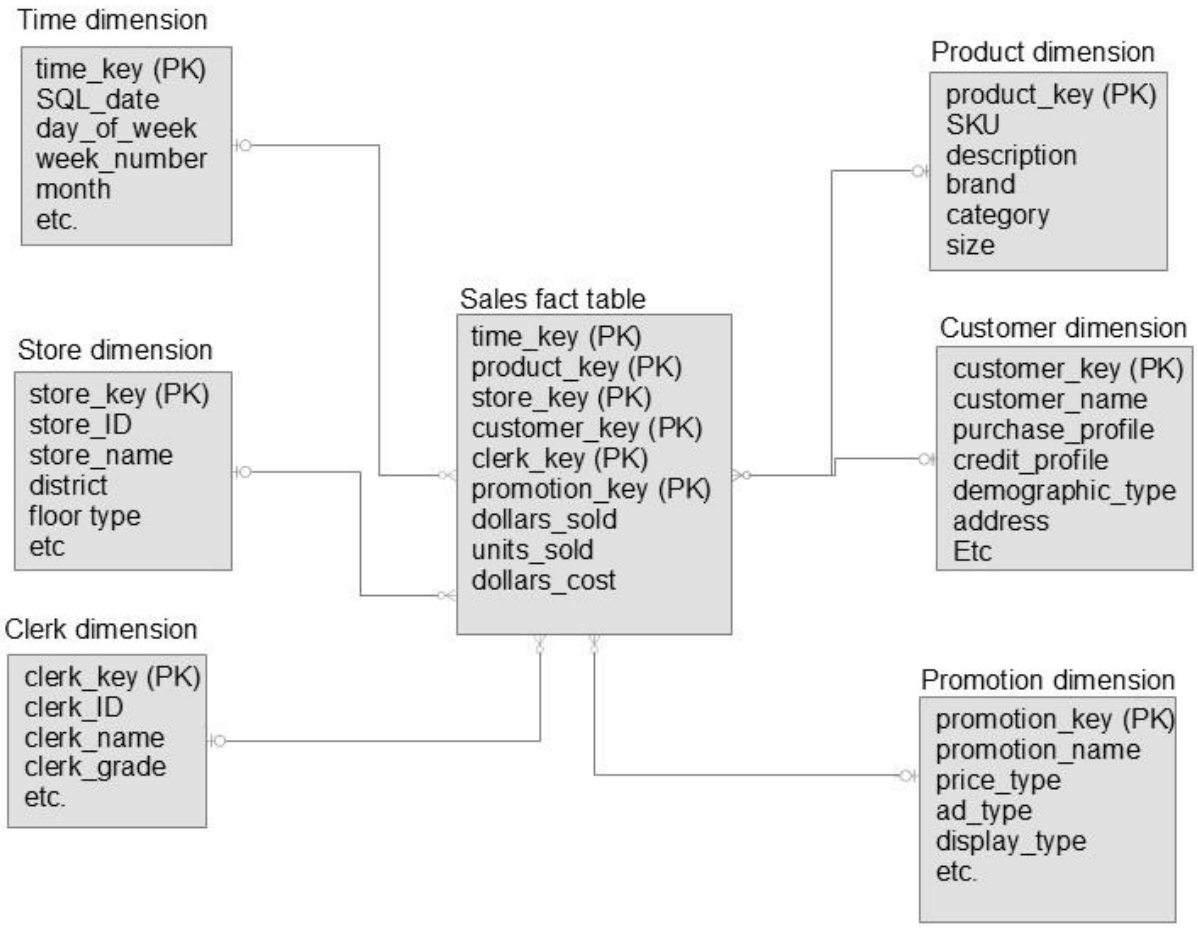


Figure 4 - 2 A retail sales process dimensional model (Kimball *et al., 1*998: 145)

As dimensional modelling focuses on only one business process at a time, there is a risk that it creates data silos for that business process. Kimball *et al.* (2008: 244) address this risk by the use of conformed dimensions and a bus architecture. Conformed dimensions are dimensions which are kept identical across all business processes and dimensional models e.g. customer dimension and hence join to multiple fact tables identically. A shrunken version of a conformed dimension may be used when less data is required at a summarised detail level however, it is imperative that the shrunken version balances back to the original version.

With the use of conformed dimensions, the overall, technology independent data warehouse bus architecture is planned. This takes the form of a simple matrix, called the “data warehouse bus architecture matrix” (Kimball *et al., 1*998: 271) which shows business processes on the rows and conformed dimensions along the columns. More recently, Kimball *et al.* (2008: 251) refer to this matrix as simply the “bus matrix”. An example of this matrix is shown in Table 4-4.

The bus matrix as depicted in Table 4-4 above is a data warehouse architectural decision that involves both the technical and business teams. Kimball *et al.* (2008: 253) emphasise the importance of support and involvement from the highest executive levels. This is because the conformed dimensions used in the bus matrix shall be the source of data for all business departments and there needs to be agreement on the data in these dimensions.

	Date	Raw Material	Supplier	Plant	Product	Shipper	Warehouse	Customer	Sales Rep	Promotional Deal
Raw Material Purchasing	X	X	X	X		X				
Raw Material Delivery	X	X	X	X		X				
Bill of Materials	X	X	X	X	X					
Manufacturing	X	X		X	X					
Shipping to Warehouse	X	X	X	X	X	X	X			
Finished Goods Inventory	X			X			X			
Customer Orders	X				X	X		X	X	X
Shipping to Customer	X				X	X	X	X	X	X
Invoicing	X				X		X	X	X	X
Payments	X				X			X	X	X
Returns	X				X	X		X	X	X

Table 4 - 4 Example of a bus matrix for a manufacturing supply chain (Kimball *et al., 2*008: 251)

From the overall architectural bus matrix, one begins to design the multiple dimensional models which make up the entire data warehouse. For this, Kimball *et al.* (1998: 272) have a four step process:

1. Choose the business process. This step may be determined by business priority. If the designers have the luxury of choice, Kimball *et al.* (1998: 272) suggest going with a business process that pulls its data from only one source.

2. Declare the grain. Granularity and grain refer to the level of detail of the data. “Declaring the grain is the equivalent to saying what is an individual fact table record” (Kimball *et al., 2*008: 273). The grain is usually chosen to be the most detailed e.g. the individual line item in an invoice instead of the entire invoice amount.

3. Identify the dimensions. These may be chosen from the bus matrix as well as choosing individual descriptive items that may be unique to this business process. The attributes that describe individual records on the fact tables are influenced by the grain that is chosen in step 2 and all dimensions should be tested against the level of detail chosen to ensure that the design will allow both the level of detail and the chosen descriptive item to link via foreign key.

4. Choose the facts. The facts are the measurable metrics that concern this business process. Each fact represents one record in a table must therefore match the grain chosen in step 2. Kimball *et al.* (2008: 248) emphasise the importance of understanding the needs of business and the available source data when making this choice.

The design of the dimensional models usually occur iteratively, going from simple conceptual designs to more concrete physical designs with constant reviewing by both business and the technical teams. Kimball *et al.* (2008: 314) encourage learning the business process as one progresses which will lead to updating the bus matrix and the dimensional models. Changes should be recorded and managed so that decisions taken are understood and remembered at a later stage.

Extract transform load (ETL)

Once the design is finalised, a source to target map should be created to plan the ETL process. Kimball *et al.* (2008: 336) recommend finalising and adding physical design detail to this map after physical design has been completed as this contributes to updating the metadata layer.

The term ETL is an industry wide term, therefore Kimball *et al.* (2008: 375) continue to refer to their defined process as such however they feel that the process has four major components: extracting, cleaning and conforming, delivering and managing. Kimball *et al.* (2008: 375) further divide these four components into thirty-four subsystems to complete the ETL system.

Extracting data involves profiling data in the early stages of a data warehouse project in order to identify useful source data. The profiled data gives an indication of data quality and volume which in turn gives an indication of storage space and time schedules needed for data utilisation. Extraction could also involve working with, working around or changing the source system's data capture method as it can be tricky to extract data while it is being captured. Kimball *et al.* (2008: 377) suggest the used of audit columns, timed extracts and database management system technologies. The data extract may be done in different formats e.g. text, xml or via ETL software that allows for data streaming. Kimball *et al.* (2008: 379) advise that data compression and data encryption may be necessary.

Cleaning and conforming data involves making changes to the data in order to make it more usable and error-free in the data warehouse thereby making it more useful to the business. Kimball *et al.* (2008: 380) assert that part of this task involves creating a commitment to error free data in source system development and source system users. The added development of this may require support from sponsors. However, the data coming into the data warehouse system will still need to be checked for errors as part of the ETL process. A system for detection and reporting of errors should be developed e.g. to detect and remove duplicates. Scheduled ETL processes should be monitored and audited, for example by row count and primary

key checks. In order to ensure that all business process report on the same data and all conformed dimensions remain identical, Kimball *et al.* (2008: 386) recommend the development of a conforming subsystem responsible for maintaining conformed dimensions.

The main purpose of the ETL system is to load or deliver transformed, cleaned and conformed data into dimension and fact tables (Kimball *et al., 2*008: 387). Source system data is constantly being changed even though some data is expected to remain static e.g. customer details. A set of rules was developed to deal with this type of data and these rules are called the slowly changing dimension types by the data warehouse industry and Kimball *et al.* (1998: 627) alike. The rules initially stipulated that all data changing in a dimension fall into one of three types. There has since been an addition of 4 other types by the Kimball Group (Ross, 2005, 2013).

As the three-type technique definition is more recognised, these are described here separately and the other four types are described briefly in the next paragraph. Type 1 data is overwritten. Type 2 data is kept and the changed data is recorded in a new row with a new key and perhaps a link to a new timestamp. Type 3 data is also kept but flagged as old and the new data is also recorded. As part of the ETL process, decisions need to be made around which type to assign to each field in slowly changing dimensions and this needs to be catered for in development of the ETL.

The newer types of slowly changing dimension patterns include type 0, 4, 5, 6 and 7. The definitions of these types are provided by Ross (2013) in a web article on the Kimball Group website. Type 0 technique caters for situations when where a dimension attribute that never changes e.g. a date of birth. Type 4 technique caters for dimension attributes that are volatile and are stored in a separate mini dimension. When history is required of the type of volatile data catered for in the type 4 technique, then a new row is added is added to the mini dimension with a field indicating which row is current. This is known as the type 5 technique. A type 6 technique involves a combination of types 1 ,2 and 3 techniques as some fields are overwritten in a row and some keep their original value. To differentiate between the

current and previous values, a current row indicator should be included. Type 7 is similar to type 6 where there are multiple rows containing different versions of the data. The difference is that each row in the dimension holds a key to a durable row containing a link to a more durable version of the record or perhaps the original version. The fact table would also contain a link to both of these rows.

Data in a data warehouse is often hierarchical e.g. a street is part of a suburb which is part of a city which is part of a province. The ETL must ensure that hierarchies that were developed remain intact. Kimball *et al.* (2008: 400) also recommend catering for data that arrives out of sync. Dimensions contain primary keys to and fact tables contain the foreign keys. As data warehouse data comes from multiple sources, it could occur that data for the fact table arrives before the dimension table data which would cause an error on most database management software systems. This type out of sync arrival should be catered for in the ETL system. Aggregates are subset or summarised dimensional models created for speedy analysis. The ETL process that loads the data warehouse should ensure that aggregates are also up-to-date. Kimball *et al.* (2008: 404) refer to OLAP cubes as an extension of the data warehouse. Many query tools exist that build OLAP cubes based on dimensional models and these cubes allow for a broader range of high-performance analysis. As such, the ETL process must also ensure that these cubes are available after ETL loads are complete.

An ETL system must work constantly, continuously and consistently. Kimball *et al.* (2008: 405) name three criteria for an ETL system: reliability, availability and manageability. These goals are achieved through many subsystems which are responsible for management of the ETL systems. One of the these subsystems is a job scheduler responsible for defining and running schedules for ETL processes, logging detailed steps of the ETL processes and notifying relevant parties of the success or failure of an ETL process or errors that might have occurred. As part of the data management, regular backups should be scheduled. This may be done as part of the ETL process. Kimball *et al.* (2008: 409) caution that ETL processes could fail due to unforeseen circumstances. In order to deal with these failures, system

restart and recovery measures must be put in place. There are often questions around data correctness from business. Kimball *et al.* (2008: 410) recommend the development of a version control system which checks back to the source system version of the data to ensure correctness. As the data warehouse gets more complex, more ETL process get added to it. This have the potential to slow down the performance and increase the errors incurred of the ETL processes. Kimball *et al.* (2008: 411) recommend the implementation of a workflow management subsystem of the ETL system which works with the job scheduler to keep track of all ETL processes by monitoring and logging measurements of hardware components affected by ETL processes, as well as measurement of software system performance which overlap with ETL processing. In this way, measures can be taken for improvement before major problems occur.

Other subsystems mentioned by Kimball *et al.* (2008: 412) which assist in management of the ETL system include a system which sorts incoming data from multiple sources as part of the transformation process; a system which tracks a data items source and dependencies; a system which handles escalation of data issues to various types and levels of support depending on severity; a system to manage data being loaded in parallel; a system to deal with security of sensitive data; a system which ensures compliance of standards and a metadata repository management system.

There is often a real time requirement for a data warehouse. This may be handled by reading directly from the source system or by using ETL processes. When an ETL system is used, there is going to be a lag between the time that data is captured and when it appears in the data warehouse. In order to reduce that lag, Kimball *et al.* (2008: 419) make the following suggestions: reading from a message queuing system instead of a batch file as this data may be formatted already allowing for quicker processing; restricting the amount of data delivered to the user thereby offering some mission-critical data in real time; updating the data warehouse directly without first extracting it to a staging area as this allows for quicker ETL process; and

allowing fact table data to be updated without immediately linking it to the most recent version of dimension data.

The ETL system is complex and development of it requires planning. Kimball *et al.* (2008: 426) emphasise the importance planning, governance and standards a when developing a large scale development like an ETL system. The ETL system is usually developed iteratively using the following high-level steps: creation of a high level source-to-target flow; choosing an ETL development tool; development of strategies for dimension management, error handling, notifications, aggregations, etc; designing of complex transformations and job sequencing; building and testing a one-time historic load process for dimensions and fact tables; building and testing the daily incremental load process.

4.3.2.7 Application track

A data warehouse may contain dozens or even hundreds of BI applications and tenfold as many users with very differing skill levels and requirements. In order to satisfy this diversity, Kimball *et al.* (2008:474) express that BI applications must adhere to the following requirements: they must be correct and accurate, they must perform well, be easy to use, be attractive and appealing to the user and applications must be properly documented, maintained and extended for long term return on investment.

Because BI application development requires the data from the data warehouse, development usually begins after the loads have completed. However, much of the BI reports requirements are defined after the requirements phase. Kimball *et al.* (2008: 507) explain that this is because the same business team who assist in requirements definition will be assisting with report requirements definition and it is efficient to define these requirements while this team have their BI thinking caps on.

Standard layout and naming is important for BI applications development as users will quickly understand the nature of the content without having to seek for

information. There will be many reports identified by users. Kimball *et al.* (2008: 512) suggest consolidating and combining similar reports into a single report and then prioritising the reports that need to be developed. BI applications are usually accessed through a portal. Kimball *et al.* (2008: 519) suggest the design of this from the first iteration as users will not need to change the entry point to the BI system at a later stage. It is imperative to review the design and development by testing and getting verification from users after each iteration and to communicate that these may change as more data gets added.

4.3.2.8 Deployment

Deployment in the Kimball life cycle is split into three parts: the technical system deployment, less technical tasks such as documentation and user training, and operational maintenance tasks.

Pre-deployment testing is advocated by Kimball *et al.* (2008: 543). This includes testing code, data quality, performance, usability and deployment scripts. A testing expert may be incorporated into the team in order to streamline this process. After pre-deployment testing has completed, the system is deployed. Kimball *et al.* (2008: 554) advise that the first deployment of a data warehouse should be the easiest as it is a deployment of a new system and gets more complicated when changes are being made to a system. Deployments should be automated via scripts. Where this is not possible, deployments should be carried out following the creation of a step by step guide which is tested on a separate test system.

Documentation needs to cater to a varying expertise of users. Dimensional models documentation, table and column descriptions may be available from metadata if the tool being implemented makes use of a metadata repository. Kimball *et al.* (2008: 559) urge developers to make use of metadata capturing of descriptions of schemas, tables, and fields in these tools as they will be available to the user. The creation of a training database is prescribed by Kimball *et al.* (2008: 562) to assist with user training and act as a searchable point for training media and manuals. Training manuals and take time to create and this time should be taken into consideration in

the entire project plan. Part of user training includes a support system for users. Kimball *et al.* (2008: 563) recommend a three tier support system. The first tier is a self-help tier where users find available training material from the available database. Tier two involves having savvy power users available to the users to assist with knowledge sharing. The power users will often have a better understanding of the business requirement behind the support question should be the first port of call for other users. When a power user is unable to assist, the data warehouse team shall be called on as a third tier of support. As this type of work can be quite demanding and time consuming, the project manager should plan to have a support team in place for this work after deployment. Kimball *et al.* (2008: 564) feel strongly that the support team come from the data warehouse team as the data warehouse team needs to remain close to the business.

When deploying a data warehouse system, it is important to have many other systems in place to keep the system secure, efficient and correct. Kimball *et al.* (2008: 567) refer to monitoring and reporting on usage of front end reports, ETL processes, and data in the warehouse so that these items can be tuned to be faster and more available. Backup and recovery of the system must be planned for from the beginning as hardware failures are unpredictable. As the data warehouse contains confidential information, security should be catered for at the database level. This is controlled by a data warehouse security manager or database administrator. In some instances, power users may be granted rights to allow or revoke access to reports and front end applications.

4.3.2.9 Maintenance and growth

Thus far, the Kimball life cycle has been insistent upon developing of standards, documentation, IT governance, communication and involvement of business users and approval and involvement from business stakeholders. As discussed in Section 3.3.2, lack of documentation and governance in the form of contracts contribute to the challenges facing maintenance developers. Section 3.3.3 mentions that Iivari (1996: 102) found that management approval is valuable when implementing tools to assist with maintenance. This goes to show that Kimball's life cycle includes key

principles in development which enhances the maintenance process. The deployment, maintenance and support phase discussed by Kimball *et al.* (2008, 541) not only focuses on maintenance and monitoring but also focuses heavily on user training and support, system monitoring and performance. However, this is not the final phase discussed. There is another phase dedicated to expanding the system. There are also maintenance concepts discussed by during the technology, data and application tracks and which form part of the development of the data warehouse.

These discussions follow this paragraph. As with the discussion on maintenance in DW 2.0, it is divided into two parts: corrective and adaptive maintenance is discussed first and then preventive and perfective maintenance is discussed.

Corrective and adaptive maintenance

Thayer and McGettrick (2007: 229) point out the that standards are a basis for software growth and foster an environment of improved communication and easier inter-changeability of developers between projects. Standards are an important theme in the Kimball life cycle. “By consistently applying standards, you’ll make it much easier for your business users and developers to navigate the complex system” (Kimball *et al., 2*008: 328). Some of the recommended standards include naming conventions, null handling, file location standards, naming of user objects and the handling of primary and foreign key objects through the use of surrogate keys which provide unique, sequenced identifiers for each row in a table.

The data model used in the Kimball methodology is the dimensional model or star schema. Kimball *et al.* (1998: 147) explain that this design allows for a predictable, standard framework, recognizable by developers and software query tools. This makes maintaining code and adding new BI applications easier. Database engine query optimizers can also make strong assumptions about the underlying models allowing for faster processing, thereby reducing the need for perfective maintenance. By using and maintaining conformed dimensions, reuse of data is streamlined thereby leading to quicker response to maintenance requests. The dimensional model also allows for graceful changes to a data warehouse as new data can easily

be incorporated by adding more columns to a fact table or dimension table without having to go through a reload process.

The two most emphasised themes in the Kimball life cycle are planning and focussing on the business requirements (Kimball *et al., 2*008: 425). Even though business requirements are given much emphasis, Kimball *et al.* (2008: 80) caution that user expectations must be managed. It is important that users understand from the onset that requirements gathering is an on-going process and that the DW system will evolve as time goes by. Keeping the user expectation realistic ensures that they are not disappointed with early iterations of the data warehouse system and that the team is not overwhelmed with maintenance and change requests.

The data warehouse architecture is described as being metadata driven. "We think about metadata as all the information that defines and describes the structures, operations, and contents of the DW/BI system" (Kimball *et al., 2*008: 116). A data warehouse tool which utilises metadata will be able to display technical metadata e.g. data structures, data sources and indexes to show connections and relationships between objects, making maintenance of these objects easier to a developer. Business metadata is descriptive and is useful to business users as it contains information of data being used, and hints on how analysis may be done. Business metadata is a form of documentation and some tools are able to generate documentation from business metadata. Process metadata contains data about business and technical process e.g. turnaround time for a sale or CPU processing usage during an ETL run. This can be used to fine tune both business and technical processes hence improving both technical maintenance and providing a return on investment.

Kimball *et al.* (1998: 728) recommend the creation of a DW steering committee to manage the maintenance and growth of the data warehouse. This team will most like contain many of the sponsors, driver and managers figures mentioned in Table 4-2; however it is wise to consider inviting other members from other areas of business. The role of this steering committee is to establish a process to prioritise the data

warehouse initiatives and establish the funding and budget for these initiatives. Maintenance requests should be prioritised and follows a standard change-request procedure, which should exist in the IT department. Major growth iterations of the data warehouse should take on the form of a full project, kicking off from the planning phase of the Kimball life cycle.

Preventive and perfective maintenance

The Kimball life cycle architecture is created with the aim of anticipating potential issues and building a system that can handle these potential issues thereby making the data warehouse more flexible and easier to maintain (Kimball *et al., 2*008: 111). To keep the architecture scalable, Kimball *et al.* (2008: 138) recommends partitioning of the presentation server. This could occur by means of running smaller individual data components on separate servers or by splitting up the same large data components over multiple servers.

Monitoring the front room architecture should take place as a preventive maintenance measure. Performance monitoring and user support monitoring create opportunities to improve response time to user queries and database indexes. Kimball *et al.* (2008: 145) reveal that whilst this may not pose an immediate need, it allows for planning for architecture upgrade as growth rates are revealed. User growth rates sourced from monitoring also provide a marketing tool to demonstrate to stakeholders a return on investment.

Kimball *et al.* (2008: 381) include preventive and perfective maintenance in the planning process. The ETL system includes plans for subsystems to handle data cleansing, quality screening, error handling and error reporting, problem escalation and parallel execution. When designing the physical database, Kimball *et al.* (2008: 344) include performance planning as part of the design. Index plans and OLAP aggregation strategies are put into place when dimensional models and OLAP databases are being designed. As part of the physical design of the back room, Kimball *et al.* (2008: 360) suggest partitioning plans, storage capacity plans and back and restore plans. Capacity plans include long term archiving and scheduled renewal

of storage media. This type of forward thinking streamlines maintenance tasks and makes budgeting for maintenance predictable.

Kimball *et al.* (2008: 405) have extended the traditional three tier extract, transform and load system to include a fourth tier to manage the ETL environment. This fourth tier carries forward the plans mentioned in the previous paragraph into the design and implementation of the ETL system. The ETL system should be constantly monitored and have processes in place to report errors and performance issues. Where possible, these should be handled automatically e.g. the early arrival of fact table data. Where a system crash occurs, Kimball *et al.* (2008: 468) recommend designing processes that allow recovery from a crash e.g. rolling back the system to a previous known good configuration. This type of management system is a preventive maintenance system. The management system also carries out perfective maintenance as Kimball *et al.* (2008: 412) recommend a sorting system and a parallel processing system to enhance the performance of the data warehouse. The ETL system is designed to maintain indexes and aggregations.

Thorough testing prior to deployment encouraged by Kimball *et al.* (2008: 544) to prevent errors and bugs, however no system is perfect and corrective and adaptive maintenance will always be required. When errors occur, Kimball *et al.* (2008: 565) advise us that these need to be communicated clearly to the user community; letting them know the nature of the problem and when it is expected to be resolved. This type of communication prevents users from working on an incorrect system.

The data warehouse is a system which is constantly growing. It is built one business process per iterative development cycle hence each iteration grows the user community. As such, there is going to be a growing number of requests for additions, changes and enhancements. Kimball *et al.* (2008: 585) reassures that “a changing system is a sign of success, not failure.” Because of the constant change, Kimball *et al.* (2008: 579) condemn handing over the data warehouse maintenance to a different team. The team should be adjusted so that front room and back room resources share the maintenance effort whilst continuing on a new iterative cycle.

4.3.2.10 Kimball life cycle summary

The Kimball life cycle is a data warehouse oriented software development methodology. It provides definitive phases and steps to go through. The major themes in the Kimball life cycle are communication with and involvement of the business and planning. It is an iterative methodology where each iteration focuses on a single business process which is called a project. The development of the entire warehouse is known as the program.

Data is modelled dimensionally in a design commonly called a star schema. This allows for better performance on queries as well as easier maintenance. The ETL system is a complex one which is not only responsible for populating the dimensional models but also managing the data warehouse with scheduled maintenance and performance tuning tasks.

Maintenance in the Kimball life cycle is largely preventive and perfective in nature. These type of maintenance activities should be planned for from initial stages of project planning and implemented as part of every release. As the Kimball life cycle is iterative, new requests are prioritised by a data warehouse steering committee that is representative of the entire business. The major data warehouse requests entail new business processes to add to the warehouse thereby causing the warehouse to continuously grow. These are identified as the next data warehouse project iteration. Minor requests are handled through the IT departments' change request process.

4.4 Challenges in data warehouse maintenance

There is not as much literature on the topic of maintenance challenges in a DW as there is on the topic of software maintenance challenges in general. This is expected as the time-period is much longer and the volume of research done is much greater than that of data warehousing as discussed in Section 1.3.

Many of the challenges experienced with transactional system maintenance, mentioned in Section 3.3.2 also apply to data warehouses. These include

documentation; working with outsourced professionals and working with large, growing systems and high volumes of data.

Data warehouses also experience challenges which are particular to data warehousing. One challenge assessing a return on investment (ROI) on an addition to the data warehouse. Kimball *et al.* (2008:27) mention that this figure needs to be estimated at the beginning of a project to convince investors of the worthiness of the project. After deployment, in order to motivate for future iterations of the project, there are many intangible benefits that need to be noted such as improved efficiency, better decision making and improved customer satisfaction. As these are intangible, they are difficult to measure.

A challenge particular to data warehouse systems is that users can often chose not to use them and continue using reports from the source systems. This is related to assessing the ROI as the ROI would be unacceptable on any system which is not going to be used. The challenge is therefore that of communicating with users and training them. “One of the main purposes of the BI applications is to provide access to the data in the warehouse to the 80 to 90 percent of the organisation who will never learn to query the data directly. Unfortunately, the remaining business users will never learn either, unless you teach them” (Kimball *et al.,* 2008: 560)

A common challenge of data warehousing is maintaining materialised view (Zhuge *et al., 1*995; Agrawal *et al., 1*997; Widom,1995). Data warehouses may be construed as a view into source systems however, since data warehouses contain actual data, the view is considered to be materialised. Materialised views are related to many maintenance challenges which are still being researched in the field.

Data warehouses are large multi-source systems. When an ETL process fails, it is often difficult to assess the failure and make corrections as it takes a long time to trace the root of the failure. The failure may occur on any of the source systems which is beyond the control of the DW team. Furthermore, due to the size of the

data, fixing the error could mean a reload of data which might require system downtime.

4.4.1 Addressing maintenance challenges

The ways to address many of the challenges experienced in data warehouse maintenance is addressed by the literature. The methodologies reviewed contain methods to detect and prevent failures, to improve performance on data loads and to monitor changes on source systems.

As DW methodologies are cyclical, many of the challenges experienced in transactional software development and maintenance should differ from those experienced in DW development and maintenance. Furthermore, as DW methodologies require more interaction with business users and executives alike, the methodologies are more communication oriented hence these challenges should also differ.

In the next chapter, we compare the challenges that are common between transactional software maintenance and DW software maintenance and examine how these can be addressed from a DW perspective. Challenges particular to DW maintenance systems are also examined further as well as methods of addressing these challenges.

4.5 Conclusion

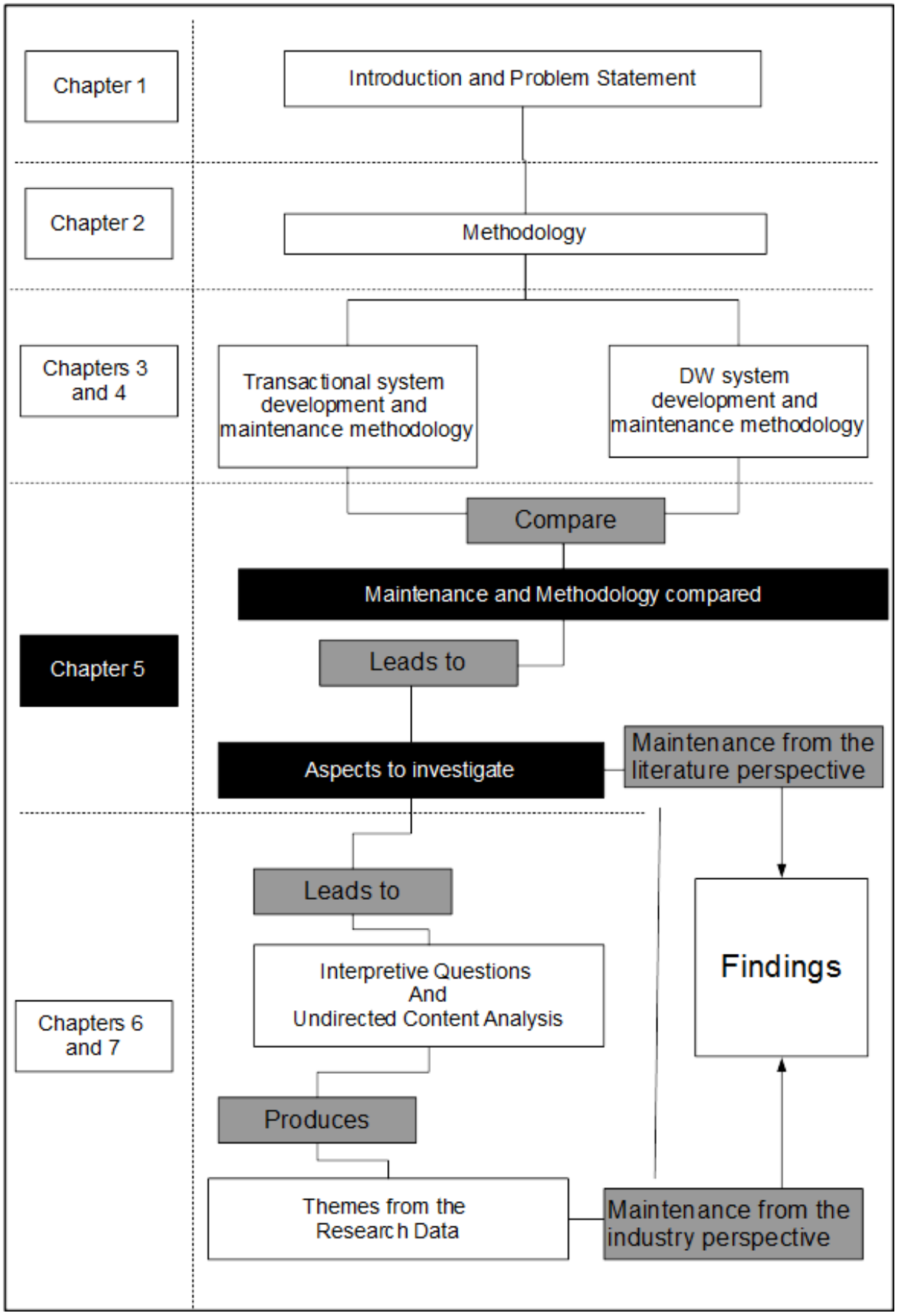
This chapter has examined data warehousing methodologies by the two pioneering authors of data warehousing, Bill Inmon and Ralph Kimball. A detailed comparison of these two author's methodologies was considered however that would detract from the purpose of this chapter which is to focus on data warehouse development methodologies and data warehouse maintenance. We will however summarise a few key points worth noting.

- Both methodologies favour iterative development cycles, as both authors agree that a data warehouse needs to be continuously worked on.
- Maintenance requests and large enhancements do not form a separate phase in either methodology; but rather a continuation into a new iteration of development.
- Much of the maintenance in both methodologies is performed behind the scenes, in the form of preventive or perfective maintenance.
- A major difference in approach is the data structure choice of the authors. Inmon favours the traditional entity relationship diagram whilst Kimball favours the dimensional model.
- The other major difference is the approach to an iterative development. Inmon recommends a Boehm's spiral methodology but provides a guiding framework to follow for data warehouse development. Kimball provides an end-to-end methodology with defined steps.

As the focus of this dissertation is on challenges in maintenance, these were challenges in data warehousing were examined in Section 4.4.

Thus far, we have covered traditional software development and maintenance in Chapter 3 and data warehouse development and maintenance in this chapter. As the focus of this dissertation is on maintenance, the next chapter compares the maintenance processes of traditional software development and data warehouses.

This page has been left blank intentionally


Chapter 1
Introduction and Problem Statement
Chapter 2
Methodology
Chapters 3 and 4
Transactional system development and maintenance methodology
DW system development and maintenance methodology
Compare
Maintenance and Methodology compared
Chapter 5
Leads to
Aspects to investigate
Maintenance from the literature perspective
Leads to
Findings
Interpretive Questions And Undirected Content Analysis
Chapters 6 and 7
Produces
Themes from the Research Data
Maintenance from the industry perspective

Chapter 5

Comparison Of Challenges Of Transactional System Maintenance And Data Warehouse Maintenance

5.1 Introduction

This research is concerned with comparing the challenges of maintenance of data warehouses and business intelligence systems with the challenges of maintenance and support of transactional software systems.

After having reviewed traditional software development methodologies and the maintenance thereof in Chapter 3; and data warehouse development and the maintenance thereof in Chapter 4, we can compare the maintenance between these topics. This chapter expands on the maintenance challenges experienced per maintenance type in both transactional and DW environments and how these challenges might be addressed when developing and maintaining a DW environment.

Five types of maintenance were listed in Section 3.3: corrective, emergency, preventive, perfective and adaptive maintenance. Emergency maintenance is not applicable in a data warehouse environment as data warehouses are rarely business critical environments. Each of these relates to a general maintenance methodology.

Before comparing different types of maintenance and maintenance methodology, a summary of the literature reviewed is provided in a sub-section of this introduction, Section 5.1.1.

General maintenance methodology is discussed in Section 5.2 where methodologies of data warehouse systems and transactional systems are compared with regard to maintenance. Thereafter, different types of maintenance are discussed in terms of transactional and data warehouse systems. These include perfective and preventive

maintenance in Section 5.3, corrective maintenance in Section 5.4 and adaptive maintenance in Section 5.5

This comparison, and a comparison of the challenges of maintenance of the two environments shall lead us to topics of directed conversation to hold with interviewees. This chapter will serve as a basis on which the interpretive research questionnaire is developed for the interview candidates of the research therefore, each section which follows contains a subsection on the types of questions which are posed during the interview process. These empirical questions are the rationale behind them are tabulated in Section 5.6. Section 5.7 concludes this chapter.

5.1.1 Methodology challenges revisited

In Section 3.3.2 and 4.4, a summary of challenges experienced in transactional software maintenance and data warehousing maintenance is provided.

As discussed in Section 1.3, data warehousing has been a research topic for a much shorter time period than has transactional software development hence there are more articles published on transactional software development. Further to this, DW systems is a subset of all software systems, hence there is bound to be many common challenges between DW systems and software systems. These challenges have been discussed in Section 3.3.2 and mentioned in Section 4.4.

The objective of this chapter is to compare challenges of maintenance between transactional systems and DW systems. The sections which follow discuss transactional systems maintenance challenges that are common to a DW system maintenance as well as maintenance challenges particular to DW systems.

The literature as reported in Section 3.3.3 refers to software evolution and software maintenance tools as possible utilities to relieve the challenges experienced in transactional software maintenance. However, these utilities are not being used

extensively in transactional software maintenance. Data warehousing methodologies are cyclical and deal with maintenance in shorter iterations. DW methodologies also include many methods for dealing with possible challenges experienced. This chapter hence examines the challenges experienced in transactional systems and then re-examines these from a DW perspective.

After examining transactional system maintenance challenges, DW system maintenance challenges are delved into. These challenges are divided between methodology, perfective and preventive maintenance, corrective maintenance and adaptive maintenance.

5.2 Maintenance methodology

In software development, maintenance is a defined phase in the methodology whereas it does not have a definitive position in a data warehouse life cycle, but rather is factored into development practices. This section contains a comparison on the challenges experienced with to these two approaches to maintenance.

5.2.1 Transactional system maintenance methodology challenges

Transaction software maintenance is cyclical but not necessarily iterative. Major maintenance requests are not handled as part of the development cycle hence this is not an iteration of development. Software maintenance as defined by IEEE Std 610.121990 (1990: 46) as work done after the project has been delivered. The maintenance cycle tends to be an addendum to actual development and implementation. The details and goal of maintenance work to be done are not known whilst in the development phase rather maintenance goals are only known after implementation.

Schneidewind (1987: 304) believes that maintenance of traditional software should be approached with a different attitude than that of being an addendum to development. He agrees with Lehman (1980: 1061) that the term maintenance should be reserved for restoring software to its previous state rather than any work

done on software after it has been delivered. Lehman pioneered the notion of software evolution which Schneidewind (1987: 304) finds to be a preferable attitude to develop on the matter of software maintenance. Whether intentionally following Lehman's model or not, it is this attitude of program evolution that underpins the notion of maintaining data warehouses.

In addition to changing attitudes toward maintenance methodology, Schneidewind (1987: 305) considers that attitudes of fixed requirements definition should also change. In sequential development methodologies, requirements are done once off, prior to development and maintenance. Once a product has been delivered, and is in a maintenance phase, additional requirements and request for changes are often made in response to usage of the system. These types of late change requests pose a problem for software maintenance as discussed by Lientz and Swanson (1981: 769). It is for this reason that Schneidewind puts forward that as new requirements emanate after deployment of software; these requirements should not be associated with a post deployment maintenance phase hence there should not be a separation of maintenance and development.

5.2.2 Transactional system maintenance methodology challenges from a DW perspective

Whilst small errors in a data warehouse may follow the regular change procedure and dealt with quickly as recommended by Kimball *et al.* (1998: 530), a major data warehouses change or request is seen as a new iteration in the development life cycle. There is a great amount of planning, testing and communication involved in each iteration. Major changes should not even be seen as maintenance per se but rather as new development project iterations. However, these iterations should not be considered as a totally new data warehouse – but rather as a development project to add to the current warehouse. In this way Kimball *et al.* (2008: 4) distinguish between a data warehouse project and data warehouse program. The program refers to the entire data warehouse whilst the project focuses on one business process pre-chosen by a data warehouse steering committee. This project runs for a finite time-span as part of the entire data warehouse program which is

ongoing and ever-evolving. Inmon *et al.* (2008: 129) also refer to the data warehouse as an ongoing program and iteration cycles as a project however the iteration cycles which Inmon *et al.* (2008: 128) refer to are sparked directly from business requests for changes and additions to the previous iteration.

This problem of changing requirements should not be as prevalent in data warehouse development. Data warehouses should be developed iteratively as recommended by Inmon *et al.* (2008: 129) and Kimball *et al.* (2008: 4). Inmon recommends use of the spiral methodology for development therefore requirements are reassessed at every iteration and development is done accordingly. Although Kimball's iterations are used to develop new business processes, there is also allowance for major changes and the choice of what to develop next is left to the business represented by the data warehouse steering committee hence there should be allowance to cater for changing requirements and new additions.

5.2.3 DW maintenance methodology challenges

Both Kimball *et al.* (2008: 585) and Inmon (2007) stress that data warehouse development continues after deployment. In this way, a traditional maintenance phase is actually another development phase. The data warehouse design is constantly added to, changed and enhanced. This poses a challenge to managers as stakeholders usually expect development team resource costs to shrink after deployment.

An environment which is constantly changing and constantly receiving more requests may be construed as being incomplete or inefficiently developed. This poses the challenge of demonstrating return on investment (ROI).

5.2.4 Addressing DW maintenance methodology challenges

In order to keep up with the constantly changing environment, Kimball *et al.* (2008: 579) strongly suggest keeping the original development team for future iterations and support on the data warehouse. The original team work closely with business as

support staff and understand business needs. This knowledge and understanding can then be incorporated into changes and enhancements made to the data warehouse.

Inmon *et al.* (2008: 129) discourage the use of the term ROI but rather suggest the term return on asset (ROA). The idea is to demonstrate to stakeholders that resources and development efforts are being reused. Each change request is not a sign of failure, but rather a sign of success that business is finding value in the data warehouse and can find more value.

Kimball *et al.* (1998: 728) approach the constant demand for change and additions to the data warehouse in a different fashion. The formation of a data warehouse steering committee is recommended. This steering committee consists of representatives from all departments of the company and their role is to prioritise data warehouse initiatives and funding. The steering committee would be educated on the iterative nature of data warehouse development and be expectant of addition and change requests. In this way, business is constantly aware of the reasons for the ongoing requests and thus made aware of the demand for more information from data warehouse. Of course, apart from this steering committee, Kimball *et al.* (2008: 3) portion out a large chunk of the Kimball life cycle to acquiring, understanding and communicating the business requirements. This should ensure a good return on investment as work being done is as per business request.

5.2.5 Empirical research questions for maintenance methodology comparison

The literature examined establishes that data warehouse maintenance should be done incrementally by the original data warehouse team, with a steering committee prioritising new requests.

Some of the interview questions listed in the following chapter focus on methodology practised when performing maintenance. These questions were posed to understand

whether these data warehouse practices are being followed, what the challenges of following these practices are and how these challenges are being addressed.

5.3 Perfective and preventive maintenance

There is a great deal of overlap in the definitions and tasks performed in perfective and preventive maintenance. As discussed in Section 3.3, Chapin (2000: 16) and Kajko-Mattsson (2000: 12) both consider preventive maintenance to be a subset of perfective maintenance. Due to this overlap, these two terms are used interchangeably in this section.

5.3.1 Transactional system preventive maintenance challenges

Of all the types of maintenance performed on transactional software systems, Singh and Goel (2007: 1) report that only 4% is preventive. They attribute this low percentage to the high pressures experienced during the development phase of software. However, Leonard *et al.* (1988: 104) discuss that the software change requests too, involve high pressures and that there is a higher likelihood of a changes being requested in the absence of preventive maintenance.

Further reasons for advocating preventive maintenance are that preventive maintenance tends to be less costly than corrective or emergency maintenance and preventive maintenance is scheduled so that software downtime occurs during periods where the software is unlikely to be needed.

However, there are factors influencing the lack of preventive maintenance. Singh and Goel (2007: 1) report these as being the attitude of managers toward maintenance, the expertise of the maintenance team and the relationship between business users and the maintenance team. Due to time and cost constraints, managers may deem preventive maintenance to be low priority. This may have the effect that an issue which may have been detected and prevented becomes evident at a later stage when costlier corrective maintenance is required. If the maintenance team is a junior one, they may not feel confident in changing software or may not

realise that preventive maintenance is required. Finally, the requirements for preventive maintenance can be difficult to understand if maintainers do not understand where possible problems could lie. This understanding is harvested through communication with business users who might have experienced a possible issue but did not feel it worth reporting formally.

5.3.2 Transactional system preventive maintenance challenges from a DW perspective

Singh and Goel's (2007:1) first concern is regarding managers' attitude toward preventive maintenance. In data warehousing development cycles discussed, preventive maintenance is not be a formal process however it is built into the overall development process. As mentioned in Section 5.2.2.1, a big challenge area in data warehouse maintenance is materialised views and the data contained therein. Kimball *et al.* (2008: 380) address this challenge in the ETL process by defining subsystems for cleaning data, logging errors and escalating errors.

Data cleansing subsystems "can be architected to create metadata used to diagnose what's wrong with the source systems" (Kimball *et al., 2*008: 380). This statement demonstrates that these systems are proactively seeking to prevent errors from occurring by building in preventive mechanisms.

Inmon *et al.* (2008: 349) also advocate perfective maintenance in the form of continuous administration of the data warehouse environment. Part of this administration includes the implementation of the near line sector and archival sector for data which is not accessed as frequently as data in the interactive and integrated sector respectively. Metadata administration is prescribed by Inmon *et al.* (2008: 352) which ensures that the business changes are continuously reflected in the data model and technical maintenance remains uncomplicated to execute as the data remains traceable between source and target. The metadata also serves to document technical and non-technical information about the underlying data.

The other concerns raised by Singh and Goel (2007: 1) are the expertise of the maintenance team and the lack of communication between business users and IT. Due to the continuous nature of data warehouse programs, Kimball *et al.* (2008: 579) advise against handing over data warehouse maintenance to a team that was not involved in the development. This also ensures that there is a strong team available to recognize the need for preventive maintenance.

As discussed in Section 5.2.1.4 above, communication with all levels of business, understanding user needs post-deployment and training of users post-deployment are part of the data warehouse development methodology. This type of communication is reinforced as the DW priorities come from a steering committee consisting of key members from all facets of the organisation. The importance of communication and training is a recurring topic in data warehouse development and is discussed further in Section 5.2.2.4.

5.3.3 DW preventive maintenance challenges

A commonly mentioned challenge with data warehouse maintenance is that of materialised view maintenance (Zhuge *et al., 1*995; Agrawal *et al., 1*997; Widom,1995). Due to the source system changes, the materialised view to the data may easily become incorrect or even non-existent. Processes, such as ETL processes need constant maintenance in order to ensure a consistent and correct view of source data.

From a preventive maintenance point of view, Do *et al.* (1998: 635) consider the challenges of very large data warehouses. Data warehouses can grow to be multiple times the size of transactional systems, posing various maintenance challenges. Materialised view reload becomes more challenging when large volumes of data need to be reloaded during small maintenance windows. Another challenge is that query retrieval becomes slower as the volume of data grows.

A data warehousing challenge which may not be naturally categorised as preventive maintenance is that of training users. Kimball *et al.* (2008: 560) rationalise that

training users allows the creation of a hierarchy of support so that power users may be able to assist other users with their queries thereby preventing unnecessary queries to the DW team. The challenging aspect of this is that the best type of training is usually developed internally whereas most companies tend to send users to solicit training from software vendors. Kimball *et al.* (2008: 560) explain that this is because users need to understand the unique characteristics of their data and environment. This type of training does require a lot of time from the DW team: Kimball *et al.* (2008: 576) estimate that it takes one day to develop an hour of training.

5.3.4 Addressing DW preventive maintenance challenges

Metadata usage by DW systems assist in addressing the challenge of changing source systems. Constant metadata comparisons makes a developer aware of changes on the source system. The DW destination tables and ETL processes should then be altered to cater for these changes. This type of metadata monitoring should be done in a proactive manner to prevent system downtime by the ETL sub-systems described by Kimball *et al.* (2008: 383). Any errors on the ETL system are monitored and reported.

The problem of Slow query retrieval is addressed in numerous ways during data warehouse development. Kimball *et al.* (2008:569) discuss monitoring both usage and performance. Performance is monitored in order to address possible areas of maintenance in a perfective fashion and usage is monitored in order to ascertain which portions of the data warehouse are may require further hardware resources in future or where further training and engaging with users may be beneficial. ETL process monitoring is also recommended by Kimball *et al.* (2008: 411) to ensure that ETL processes are not getting too slow to be easily maintainable.

Kimball *et al.* (1998:147) advocate the use of dimensional modelling when developing a data warehouse. They reason that dimensional models perform faster joins, are easier to understand and are more predictable to a database management system and developers alike. This way, querying dimensional models in unexpected

ways do not result in slower query response time as one would expect from entity relationship modelling. Furthermore, dimensional models have become an industry standard, allowing for predictable handling of changes in data using slowly changing dimension rules and ETL tools designed to load dimensional models.

In the DW 2.0 framework, Inmon *et al.* (2008: 203) prescribe that data should flow between different sectors of the data warehouse. For cost effectiveness, these sectors reside on different hardware. Frequently accessed data is placed in the interactive sector which resides on high performance hardware. Data usage is monitored and data with low access rates are moved out of the interactive sector into the integrated sector and data with much lower access rates are moved to the archival sector. In this way, the volume of data in the interactive sector remains manageable and easily maintainable.

Two ways in which the challenge of training users can be addressed is through better communication during the DW project and better use of metadata. There is constant interaction and communication between business users and IT during the Kimball life cycle. Kimball *et al.* (2008: 580) recommends spending time with users to learn how the data warehouse system is being used as well as the formation of the data warehouse steering committee consisting of influential leaders from business. The steering committee would be responsible for providing budget and resource planning for the data warehouse. As such, the inclusion of important non-technical aspects such as training is prioritised. As part of and the planning phase, Kimball *et al.* (2008: 43) advocate the development of a communication plan which includes internal team communication, sponsor debriefings and communication with the user community. As part of the deployment phase, Kimball *et al.* (2008: 562) advise on the creation of a training database which will assist in users training themselves and relying less on the DW team.

Inmon *et al.* (2008: 357) also acknowledge that end-user relationships need to be managed. Training and education endeavours such as facilitated training sessions, help-desks and newsletters are suggested to assist in building relationships with end

users. In order to ensure continued levels of availability and performance from the data warehouse, Inmon *et al.* (2008: 357) support the use of a service level agreement (SLA). An SLA also assists in encouraging early feedback from users regarding issues on the data warehouse thereby preventing problems from escalating into larger and costlier ones.

Kimball *et al.* (2008: 558) reason that metadata can expedite documentation creation which assists users in quickly understanding the DW system. However, business metadata is not the only type of metadata consumed by users. Watson and Haley (1998: 36) explain that users require more process oriented information and technical than on a transactional system. Examples of this information include: the last data refresh time, the source system for the data and calculations pertaining to the data. By extensively capturing metadata and making it available to users in an intuitive format, the DW team can prevent having to spend time on answering many support and maintenance queries.

5.3.5 Empirical research questions for preventive maintenance comparison

Preventive maintenance is often neglected in transactional systems. While this should not be so with data warehouses, there are other challenges to preventive maintenance. The challenges surrounding preventive maintenance in data warehousing include the attitude toward preventive maintenance, the volume of data, dimensional modelling, user education and the use of metadata for documentation and user education. As such, questions are posed which aim to understand how these challenges are being addressed practically.

5.4 Corrective Maintenance

As discussed in Chapter 3, corrective maintenance is performed in response to software failure.

5.4.1 Transactional system corrective maintenance challenges

It is reported that much effort is spent in program comprehension during maintenance (Von Mayrhauser & Vans, 1995: 44; Ko, Aung, & Myers, 2005: 126). The difficulties of program comprehension are amplified when compounded with lack of documentation and maintenance developers who do not have prior knowledge of the system being maintained as denoted by Lientz (1983: 274).

Lethbridge *et al.* (2003: 36) have found that the task of documentation is unlikely to be performed unless it is formally required as part of the change process yet documentation was found to be useful and valuable by software engineers.

Bhatt *et al.* (2006: 1) have investigated outsourcing maintenance as this a common practice where development resources are limited. One of their findings is that there is a strong correlation between time that an outsourced maintenance team enters a project and success of the maintenance. As knowledge of a system is gained during development, maintenance is more successful when done by a team who have entered the project during a development phase. Another finding by Bhatt *et al.* (2006: 4) is the greater the depth of system knowledge that the maintenance team has, the better the relationships is with the client hence there is better communication with end users.

5.4.2 Transactional system corrective maintenance challenges from a DW perspective

Due to the ongoing nature of a data warehouse project, Kimball *et al.* (2008: 574) recommend keeping on the same number of developers for maintenance and support. If the initial project was a small one, the recommendation is to add on more resources. Further to this, Kimball *et al.* (2008: 579) advise against handing over support and maintenance to a separate maintenance organization. Keeping the same developers on the project during maintenance will reduce the time required to gain understanding of the data warehouse system in order to maintain it.

Another reason program comprehension may not pose as great a challenge in data warehouse systems as compared to transactional systems is that the underlying data is likely stored in a dimensional model. One of the strengths of dimensional modelling as pointed out by Kimball *et al.* (1998: 147) is that it is predictable and more understandable than entity-relationship models. This allows a developer to make assumptions about the underlying data design. This attribute of dimensional models is especially advantageous in adaptive maintenance which is discussed in the next section.

Data warehouses tools store technical metadata and allow for the capture of business metadata as well as editing of technical metadata. As mentioned, this metadata can be used in the creation and maintenance of source to target maps which assist in tracing failures in a complex data warehouse system. Kimball *et al.* (2008: 576) explain that metadata queries and reports can be used to supplement technical documentation and business documentation, thereby making the documentation tasks and user education tasks easier. Knowledgeable users provide valuable feedback when failures occur which could also assist in expediting remedial action.

5.4.3 DW corrective maintenance challenges

“Failure can happen anywhere, from populating the data warehouse, to refreshing multi-dimensional databases, to processing end-user queries; each task is long and resource intensive, and each task is costly to roll-back and restart” (Do *et al., 1*998: 633). These types of failures pose the challenge of tracing the root of the failure which involves tracing a data item back to its source for comparison and to ensure correctness. Inmon (2005: 391) explains that data may be transformed on the way from the source, there may be a combination of sources and the names of data fields may change making this a difficult task.

5.4.4 Addressing DW corrective maintenance challenges

Source to target maps make the task of tracing the data easier. Kimball *et al.* (2008: 336) encourages documenting source to target maps during the analysis and design process and adding more detail to these during the development process. As source

to target documentation is part of the design and development process, developers make use of it hence they are more encouraged to create and revise this type of documentation as needed.

Although source to target maps are usually drawn up as part of the data warehouse design process, they need to be continuously maintained as the DW system changes. Inmon *et al.* (2008: 97) explains that ETL metadata is updated as changes between source, targets and transformations occur. This updated metadata makes documentation such as source to target mapping easier to maintain as metadata can be used in the creation of documentation.

5.4.5 Empirical research questions for corrective maintenance comparison

Dimensional modelling and metadata should assist in addressing the challenge of understanding and maintaining data warehouse systems. Unlike transactional systems, there is no maintenance phase in data warehouse development. The literature studied recommends that the original development team continue with further development and maintenance. Questions were posed during interviews to understand the practice of data warehouse development in this regard.

5.5 Adaptive maintenance

Adaptive maintenance has some overlap with corrective maintenance. Von Mayrhauser and Vans (1995: 45) have tabulated the tasks involved in different types of maintenance indicating that both corrective and adaptive maintenance involve the time consuming process of program comprehension as discussed in Section 5.2.3 above. In addition to this, adaptive maintenance also involves responding to multiple new user demands.

5.5.1 Transactional system adaptive maintenance challenges

Lientz and Swanson (1981: 769) report that these user demands are seen as the most severe overall maintenance problem. The list of tasks that Von Mayrhauser and

Vans (1995: 45) tabulated for adaptive maintenance includes requirement gathering, analysis and design. Lientz (1983: 274) discusses that this type of maintenance should be encapsulated in a new development cycle however a study undertaken by him was unable to conclude that this is the case. A possible reason that adaptive maintenance may not being treated as a project, as pointed out by Bhatt *et al.* (2004: 4) is that project tasks for maintenance may be met with resistance from managers who might place priority on new development and have unrealistic perceptions of target dates and time required for maintenance tasks.

5.5.2 Transactional system adaptive maintenance challenges from a DW perspective

Additional user requests should not be regarded as a problem in a data warehouse system. Both Kimball *et al.* (2008: 585) and Inmon *et al.* (2008: 111) emphasise that a data warehouse should be built to be adaptable to business change. The prioritization and budgeting of unplanned requests should be handled by a steering committee which Kimball *et al.* (1998: 728) recommend consist of influential leaders from the business community. It is emphasised that the members of the steering committee understand the enterprise-wide benefits of using the data warehouse to ensure the best prioritization of data warehouse development and maintenance activities. Furthermore, as the steering committee consists of influential members of the organisation, data warehouse projects ought to get the budget, time and resources required. Another way that Kimball *et al.* (2008: 248) address the challenge of continuous user requests is by planning a data warehouse development one business process at a time therefore the business understands when different subject areas will be included. Each business process is added to the data warehouse through a new project cycle. This correlates with the recommendation made by Lientz (1983: 274) for adaptive maintenance.

A more technical resolution to user change requests is proposed by Inmon *et al.* (2008: 118). This involves keeping data which is more current on separate hardware. As change requests would usually only affect more current data, only this hardware would be affected by change request. As such, other parts of the data warehouse

can continue to be accessed unaffected. Furthermore, if data reloads are required for the changes, they can be effected more swiftly as the subset of data shall be smaller.

The Kimball life cycle also has technical advantages to adaptive maintenance. Kimball *et al.* (1998: 148) explain that the dimensional model is gracefully extensible as additional data requests can be accommodated by adding new rows or columns to existing tables without needing to change existing program code or reload entire data sets. Furthermore, new dimensions can be added to a star schema which immediately caters for more ways to analyse a process without affecting the current analysis.

5.5.3 DW adaptive maintenance challenges

Another maintenance problem associated with materialised views is when the source system data schema changes due to maintenance development on source systems or legacy systems connected to a data warehouse. Inmon (2005: 181) relates that when a legacy system is changed, there is a ripple effect to connecting systems. Rundensteiner *et al.* (2000: 59) explain that a data warehouse typically connects to multiple sources. Some of these sources may not be mature systems hence their database schemas are being improved upon. This poses a problem to a data warehouse which reads from these sources as the code which extracts data from these sources will no longer work.

A DW maintenance challenge is adapting to the change of time-sensitive data in the source systems which normally remains static e.g. address of customer. Address information is important in demonstrating geographical relationships. When a person changes address, the new address no longer holds significance to past relationships. The challenging issue is reporting and analysing historical data and current data at the same time.

5.5.4 Addressing DW adaptive maintenance challenges

A simple solution to the problem of development occurring on source systems could be communication of changes between departmental IT teams which would lead to changes being implemented on the data warehouse extract layer when the data source changes. This may be part of a company's change management protocol. Unfortunately, data sources may be housed externally and communication between parties is not always viable. In these instances, source system and ETL monitoring and escalation systems as described by both Inmon *et al.* (2008:174) and Kimball *et al.* (2008:376) would assist in ensuring that source change does not result in system failures and that the maintenance team's attention is drawn to problematic source connection. Maintenance of these problematic systems is usually expedited with the use of a metadata repository which allows for quick extract definition changes.

The use of metadata also assists with adaptive changes in response to changing source systems. Inmon (2005: 184) explains that metadata keeps track of data connections and transformations that occur allowing information to be traced back from the data warehouse to source systems. Kimball (2008: 117) explains that when changes are made to a source system, it is only the metadata that needs to be updated for an ETL program to cater for those changes. These changes may appear seamless to the user community.

In the previous paragraphs, both possible solutions to the schema change require human intervention, and some development and implementation time. Rundensteiner *et al.* (2000: 59) propose a system called Evolvable View Environment (EVE) as a dynamic, automated solution to the materialised view problem. EVE is described as middleware as it is logically placed between the source systems and the data warehouse. All source systems notify EVE of connection details and send updates to EVE when changes occur. When this occurs, EVE automatically updates the SQL code used to cater for schema changes. EVE has been successfully implemented as a prototype as discussed by Lee *et al.* (2002: 949).

A possible way of addressing the issue of time sensitive data that changes is implementing a slowly changing dimension type solution defined by Kimball *et al.* (1998: 627). The slowly changing dimension types were discussed in Section 4.2.2.6. The idea is that the business needs to decide on whether or not historical data is valid for a particular data set. If so, keep a copy of the historical data and if not, overwrite the historical data.

5.5.5 Empirical research questions for adaptive maintenance comparison

Similar to corrective maintenance, challenges of adaptive maintenance include understanding of the system to be maintained. Hence, the questions posed on metadata and documentation is applicable to adaptive maintenance as well. Further to these questions, questions were posed regarding dimensional modelling, separation of data which should assist with maintaining higher data volumes and prioritization of user requests.

5.6 Summary of DW and OLTP maintenance literature

In this chapter the comparison between transactional system maintenance and data warehouse maintenance has been broken down into four major empirical research areas:

1. Maintenance methodology
2. Perfective and preventive maintenance
3. Corrective maintenance
4. Adaptive maintenance

Each of these research areas were analysed in terms of challenges experienced and addressing those challenges to understand the empirical research aspects pertinent to these research areas from a DW maintenance perspective. The empirical research areas above were therefore broken down into:

- Transactional system challenges
- Transactional challenges from a DW perspective
- DW system challenges

- Addressing those DW system challenges

Table 5-1 on the following page summarises the key points discussed and the last column represents the empirical questions that emerged from the literature for each research area.

In Chapter 6, the research conducted is presented and the development of the interview questions based on the literature reviewed this chapter and previous. The summary in Table 5-1 demonstrates the rationale behind the development of these questions. The last column contains the aspects to be investigated in the interpretive research questionnaire. These aspects are numbered in order to reference them later. Aspects that were repeated in the table kept their original number.

In order to assist with readability, these aspects are listed in order of appearance below:

1. Cyclical or iterative methodologies allow requirements to be added in new iterations.
2. Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.
3. Use of a steering committee (which includes representation from the source systems, business management and IT)
4. Priority of preventive maintenance and when it is performed.
5. Use of metadata including documentation.
6. Use of dimensional models.
7. Addressing challenges of high volumes of data.
8. Partitioning scheme usage and strategy.
9. Training of users and cross skilling of developers.
10. Is maintenance being done primarily by junior developers?
11. Use of outsourcing.
12. Involvement of development team on maintenance tasks.
13. Logical hardware separation.
14. Usage and availability of maintenance tools.

Research area	**Transactional system challenges**	**Transactional challenges from a DW perspective**	**DW system challenges**	**Addressing DW challenges**	**Aspect to be investigated**
Methodology	• Major maintenance requests are not handled as software development iterations. • Software does not evolve naturally but is phased. • Having one requirements phase is too rigid in a changing environment.	• There should not be a maintenance phase for major requests but a new iteration. • Because of iterative development, changing requirements should not be as noticeable as delivery phases are shorter.	• Managing the expectations that costs should shrink after deployment. • A good DW is always receiving more requests which may be construed as it being incomplete.	• Important that the original development team stays on for future iterations. • Communication with stakeholders through the formation of a steering committee.	1. Cyclical or iterative methodologies allow requirements to be added in new iterations. 2. Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance. 3. Use of a steering committee.
Preventive or Perfective maintenance	• During software development phase, high pressure and tight deadlines causes neglect on preventive maintenance. • Lack of preventive maintenance could result in higher number of requests for corrective maintenance. • Preventive maintenance takes a lower priority with managers. • Maintenance teams are often inexperienced. • Communication is lacking between business users and developers.	• Preventive maintenance should be built into the methodology and not be a separate task. • ETL systems should proactively check for errors. • Preventive maintenance occurs through continuous administration e.g. archiving and aggregation. • Metadata administration ensures changes are reflected. • DW methodologies emphasise continuous communication with business.	• Maintenance of materialised views is difficult as source systems change, often without notice to DW team. • Volumes of data are much larger in DW environment. • Training of users to become super users creates a hierarchy of support.	• High volumes of data addressed through partitioning schemes. • Training of super users to create a hierarchy of support. • Dimensional models assists in addressing performance concerns, user training concerns and developer cross skilling. • DW methodology encourages user input from the beginning which promotes user training. • Updating of metadata assists in quick redefinition of materialised views • Metadata expedites documentation creation.	4. Priority of preventive maintenance and when it is performed. 5. Use of metadata including documentation. 6. Use of dimensional models. 7. Addressing challenges of high volumes of data. 8. Partitioning scheme usage and strategy. 9. Training of users and cross skilling of developers. 10. Is maintenance being done primarily by junior developers? 1. Cyclical or iterative methodologies allow requirements to be added in new iterations. 3. Use of a steering committee.

Research area	**Transactional system challenges**	**Transactional challenges from a DW perspective**	**DW system challenges**	**Addressing DW challenges**	**Aspect to be investigated**
Corrective maintenance	• Understanding the program being maintained. • Due to outsourcing, maintenance tends to handed over to a different team. • Documentation creation is not sufficient.	• DW methodology advises against a different maintenance team. • Dimensional models make program comprehension easier. • More emphasis is placed on metadata in DW systems. This allows for faster documentation.	• Because of multiple sources, there are more points of failure. • Reloading data after load failure takes time making load errors and failures more costly.	• Documentation in the form of source to target maps allow for quick tracing of data. • Keeping metadata updated fosters the maintenance of documentation.	5. Use of metadata toward documentation. 11. Use of outsourcing. 2. Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance. 6. Use of dimensional models. 7. Addressing challenges of high volumes of data. 12. Involvement of development team on maintenance tasks.
Adaptive maintenance	• Understanding the program being maintained. • User demands for new functionality which should be encapsulated in a development cycle rather than maintenance. • New development takes priority over adaptive maintenance.	• DW system usage encourages new requests from users. • There should be a steering committee involved in prioritisation of requests. • A DW is build one business process at a time hence additions are being made constantly. • If data physically separated according to usage requirements, then the entire system does not need to be altered for a change. • Dimensional models are gracefully extendable.	• When source systems change, the changes have ripple effects to the DW system. • Updating time-sensitive data e.g. customer address which normally remains static.	• Source changes need to be communicated to DW team. • ETL monitoring and preventive measures should be in place in the case of failure on the source system. • Metadata keeps track of connections and transformations hence when source changes occur, only metadata needs to be updated. • Implementation of automated maintenance tools can assist with changes. • Defined slowly dimension types allow for handling of change to time sensitive data.	5. Use of metadata toward documentation. 6. Use of dimensional models. 13. Logical hardware separation. 14. Usage and availability of maintenance tools. 8. Partitioning scheme usage and strategy. 3. Use of a steering committee. 4. Priority of preventive maintenance and when it is performed. 1. Cyclical or iterative methodologies allow requirements to be added in new iterations.

Table 5-1 Summary of empirical research areas

5.7 Conclusion

This chapter has undertaken a comparison of maintenance in transactional systems and maintenance in data warehouse systems. As maintenance challenges for transactional systems have previously been discussed, the maintenance challenges being faced in data warehouse development were discussed. These included both technical and non-technical challenges.

Many of these non-technical challenges are addressed through the methodologies of data warehouse development. There is a difference in the attitude toward maintenance of a data warehouse and that of transactional systems. With transactional systems, major requests for additions to a system are viewed as maintenance as such these requests are separate from development. Maintenance requests follow a maintenance cycle which is an addendum to a development cycle. The transactional system maintenance cycle has a different team and budget associated with it. In a data warehouse development, major maintenance requests undergo a full data warehouse development cycle with the same development team budgeted for by the data warehouse development budget.

In transactional systems, preventive and perfective maintenance are often neglected. This may be due to time, budget and resource constraints as well as a lack of understanding by project managers. This neglect leads to the need for costlier corrective action at a later stage. Data warehouse development practices incorporate preventive and perfective maintenance measures into the development methodologies. Aside from these technical measures, data warehouse development practices also encourage a greater deal of communication with business stakeholders and end users. This allows for better communication between business and the data warehouse team, resulting in early reporting and handling of performance issues.

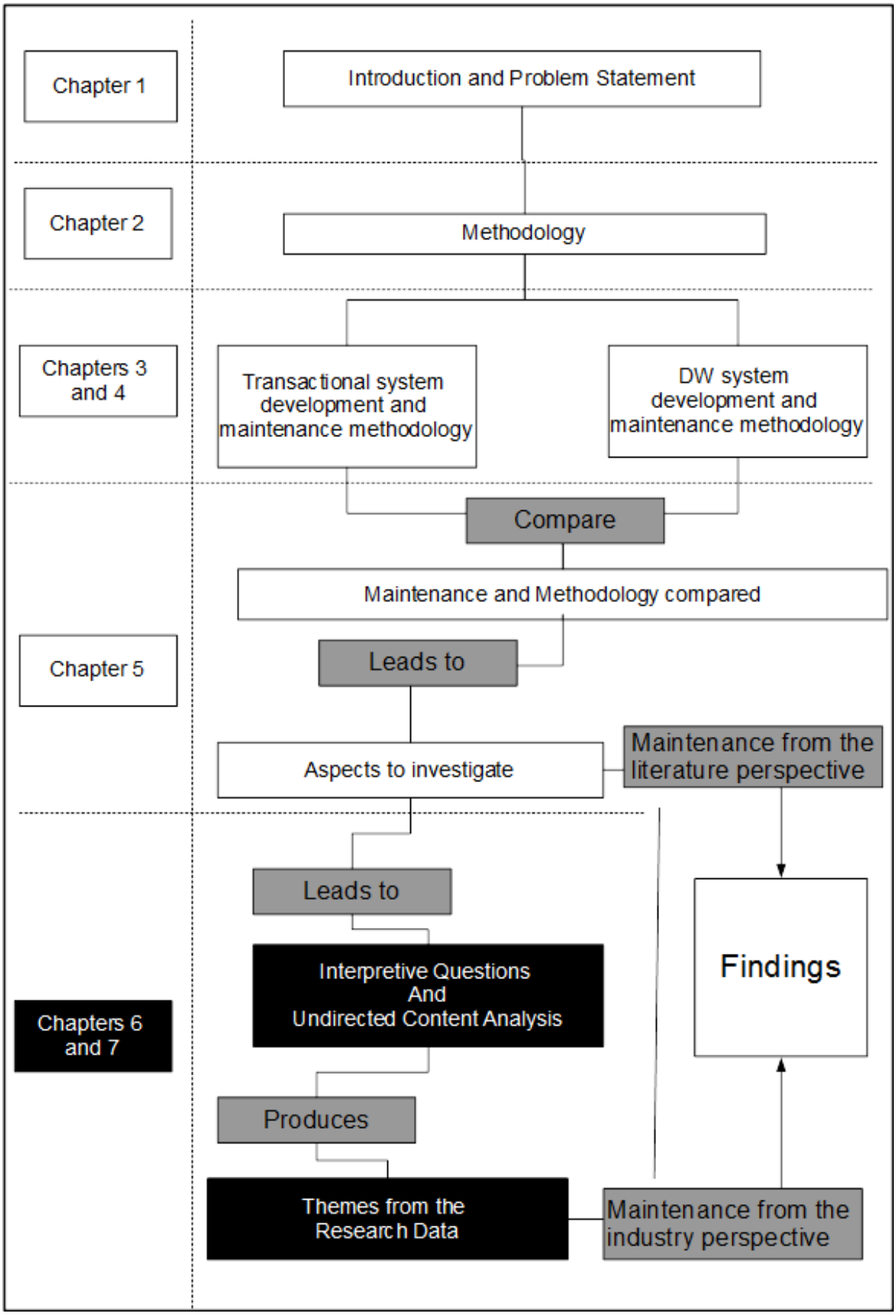
A significant challenge faced with corrective and adaptive maintenance of transactional systems is program comprehension. This challenge is compounded with lack of documentation and a maintenance team which does not include members from the original development team. Data warehouse development is

iterative and ongoing; hence there should not be a change between development and maintenance teams. Data warehouse systems are often modelled using dimensional modelling which is intrinsically more intuitive and allows for easier comprehension of data warehouse systems. The use of metadata allows for automated construction of documentation. Metadata also assists in tracing data back from the data warehouse to the source system which results in easier fault finding.

Adaptive maintenance in transactional environment faces the challenge of ongoing user requests to a deployed system. In a data warehouse system, these requests should not be regarded as a challenge as the data warehouse is never considered to be fully deployed. As an organisation changes, so too will the data warehouse. User requests for changes are implemented through a new development project cycle. The use of dimensional models allow for graceful extensibility to a data warehouse because more data can be added to a data warehouse without a ripple effect of changes being required.

In the next chapter, empirical research is conducted and discussed. The comparison performed in this chapter serves as a basis for questions to interview data warehouse developers with.

This page has been left blank intentionally


Chapter 1
Introduction and Problem Statement
Chapter 2
Methodology
Chapters 3 and 4
Transactional system development and maintenance methodology
DW system development and maintenance methodology
Compare
Maintenance and Methodology compared
Chapter 5
Leads to
Aspects to investigate
Maintenance from the literature perspective
Leads to
Findings
Chapters 6 and 7
Interpretive Questions And Undirected Content Analysis
Produces
Themes from the Research Data
Maintenance from the industry perspective

Chapter 6
Interpretive Research Conducted

6.1 Introduction

The overarching objective of this research is to compare the challenges of maintenance of data warehouses and business intelligence systems with the challenges of maintenance and support of transactional software systems. In this chapter, this objective is pursued empirically. This was done by gathering data and analysing it and drawing conclusions from it. The purpose of this chapter is to gain an understanding of the maintenance practices used by DW practitioners in South Africa.

In the previous chapter, maintenance between transactional systems and data warehouse systems were compared. This comparison was broken down into maintenance types. Section 6.2 summarises the previous chapter and the comparison that it makes. The question set for participants was based on this comparison. The researcher was interested to understand whether the inherent strengths of DW methodologies in terms of maintenance as discussed in Chapter 5 are utilised by DW professionals.

Data was gathered though an interpretive questionnaire which was either answered in written format or verbally. The development of the questionnaire is discussed in in Section 6.3 and the data gathering process is discussed in Section 6.4.

In Chapter 2, it was decided to use content analysis as data analysis methodology.. Section 6.5 breaks this down further into a type of content analysis, research analysis mode and research analysis method. The coding process was an iterative one which took three phases to complete. These phases are also described in Section 6.5.

In Section 6.6, the coded data are analysed and conclusions are drawn in subsections. Section 6.7 summarises the chapter.

6.2 Summary of DW and OLTP maintenance literature

In Section 5.6, empirical research aspects where extracted from the literature which was surveyed. These aspects were numbered for reference purposes. Table 6-1 contains these numbered aspects

1	Cyclical or iterative methodologies allow requirements to be added in new iterations.
2	Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.
3	Use of a steering committee (which includes representation from the source systems, business management and IT)
4	Priority of preventive maintenance and when it is performed.
5	Use of metadata including documentation.
6	Use of dimensional models.
7	Addressing challenges of high volumes of data.
8	Partitioning scheme usage and strategy.
9	Training of users and cross skilling of developers.
10	Is maintenance being done primarily by junior developers?
11	Use of outsourcing.
12	Involvement of development team on maintenance tasks.
13	Logical hardware separation.
14	Usage and availability of maintenance tools.

Table 6-1: Empirical research aspects

6.3 Interview development

Table 5-1, which presents a summary of Chapter 5, was used as the principle guide to the development of the interpretive questionnaire. The questions were also influenced by literature presented in other previous chapters. As a result, the issues

identified in Table 5-1 are not limited to the theme in which they were first raised. For example, the use of dimensional models came only to the fore as part of "preventative and perfective maintenance" but it is in nature a methodological issue. Table 6-2 contains the questions that were posed to participants. The underlined word under the column "research area" refers to the empirical research area that the question emanated from in Table 5 – 1. The order of the questions in Table 6-2 do not correspond to the order of aspects of investigation noted in Table 5-1 as the questions were ordered to ensure a logical flow for the participants.

Each question in Table 6-2 is accompanied by an aim or theme and supporting reference. In general, the aim of the interview questions is to understand how maintenance is done by practitioners in the DW environment. Many questions were open ended and many questions dealt with topics that are pertinent to multiple empirical research areas from Table 6-1. This is because questions were formulated to try learn as much from participants as possible without steering them toward an answer. Most questions were open ended e.g. question 3 (Q3) to allow a broad range of responses from the participant. Hence, there are many questions which contain the word "overall" under research area. This indicates that the question covers multiple areas. Others questions were more specific e.g. Q12 and Q13.

	Question	Research area	Aspect	Reference
1	Describe the methodology or development process that you go through when developing data warehouses and where maintenance fits into this methodology.	Methodology/Overall Maintenance Understand whether Iterative development is being used (Aspect 3). Understand whether maintenance is being done as a separate phase or if maintenance is being done as a new iteration (Aspect 3). Understand whether dimensional modelling is being used (Aspect 6).	1 6	Kimball *et al.* (2008:4) – program and project iteration. Inmon *et al.* (2008: 127) - cyclical development.
2	In particular, describe the process of making additions to the data warehouse after initial deployment.	Methodology/Overall maintenance Follow up on question 1. In particular, if not touched on in question one, understand whether a steering committee is being used to decide on what is going to developed next (Aspect 1). Understand whether a different team is doing maintenance	3 2	Steering committee - Kimball *et al.* (1998: 728).

	Question	Research area	Aspect	Reference
		(Aspect 2).		
3	Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced.	Methodology – General: Open ended question to understand whether the experience/challenges mirrors that of the literature (Multi Aspect). Examples include: • Is materialized view maintenance a problem (Aspect 5)? • How are slowly changing dimensions being dealt with (Aspect 6)? • Do high volumes pose a challenge (Aspect 7)?	MULTI, e.g. 5 6 7	Slowly changing dimension defined by Kimball *et al.* (1998: 627).
4	How do you deal with handling and prioritization of on-going requests?	Adaptive maintenance/ Methodology. Are new requests being handled and prioritized by use of a steering committee (Aspect 1)? Are cyclical development methodologies being used (Aspect 3)?	3 1	Inmon *et al.* (2008: 127) – spiral methodology. Kimball *et al.* (2008: 4) – project and program. Steering committee - Kimball *et al.* (1998: 728).
5	How do you address the challenge of working on a data warehouse that you have not worked on before?	Overall Question, Corrective/Adaptive maintenance Outsourcing data warehouse development often leads to an in-house team taking over for maintenance, or in a similar fashion, many companies change to a more affordable and less experienced outsourced team for maintenance after a system is developed (Aspects 2, 11). Changing DW team for maintenance (Aspect 2). Use of documentation and metadata (Aspect 5). Ease of understanding of dimensional models (Aspect 6).	2 11 5 6 12	Kimball *et al.* (2008:579) advise against handing over support and maintenance to a separate maintenance organization. Kimball *et al.* (1998: 147) – dimensional model. Kimball *et al.* (2008: 336) – use of source to target maps. Kimball *et al.* (2008: 558) use of metadata for documentation. Inmon *et al.* (2008:359) – managing consultants.
6	Describe the use of documentation in maintaining and supporting data warehouses.	Corrective maintenance, Adaptive maintenance Follow up to Q5 as the open ended nature of Q5 might not have elicited a response which considered documentation (Aspect	5	Kimball *et al.* (2008: 336) – use of source to target maps. Kimball *et al.* (2008: 558) use of metadata

	Question	Research area	Aspect	Reference
		5).		for documentation.
7	Describe when and how documentation is created.	Methodology Follow up to question 5/6. Is documentation mandatory per phase? Is documentation kept current? Is metadata used for documentation (Aspect 5) ?	5	
8	Describe the relationship that DW team has with business users and stakeholders and the interaction that goes on between the DW team and the business users and stakeholders.	Overall Question, Methodology, Corrective/Adaptive maintenance To determine whether there is close interaction between DW team and business and if there is a communication plan (Aspect 9). To establish the type of training that goes on. To determine whether there is a steering committee which prioritizes new developments (Aspect 1).	3 9 12	Steering committee - Kimball *et al.* (1998: 728). Kimball *et al.* (2008: 43) - communication plan. Kimball *et al.* (2008: 584) - Training and support. Marketing the data warehouse. Inmon *et al.* (2008: 357) - "One of the most important jobs of management of the DW 2.0 environment is that of managing end-user relationships and expectations"
9	How are the end users supported?	Follow up to question 8. A steering committee made up of executives shall encourage the use of end user support from the DW team (Aspect 1). User training is part of end user support (Aspect 9)	3 9	
10	Describe the regular maintenance tasks that are performed on the data warehouse environment.	Preventive maintenance Establish whether preventive/ perfective maintenance is being performed as recommended (Aspect 4).	4	Inmon *et al.* (2008: 349) – continuous administration of the data warehouse. Kimball *et al.* (2008:380) - Subsystems.
11	Are there maintenance tasks that you feel should be included and if so, what factors are influencing the implementation of these?	Preventive maintenance Establish influencing factors of on-going preventive maintenance (Aspect 4) e.g. manager support, lack of communication with users, lack of experienced maintenance team (Aspect 10), outsourced	MULTI 4 10	Kimball *et al.* (2008:43) advocate the development of a communication plan. Kimball *et al.* (2008:580) recommends spending

	Question	Research area	Aspect	Reference
		maintenance team. This is a multi-aspect question.		time with users to learn how the data warehouse system is being used as well as formation of a steering committee.
12	Talk about your experience with meta data use practically in the maintenance of data warehouses and the other practical uses of metadata.	Overall Question Understand if metadata is being used or is it still largely left unused (Aspect 5). Understand if business metadata being captured (Aspect 5).	5	Kimball *et al.* (2008:576) explain that metadata queries and reports can be used to supplement. Inmon (2005: 392) - tracing data back to the source.
13	Describe the advantages or disadvantages of maintaining dimensional models.	Overall Question Are developers taking advantage of graceful extensibility, ease of understanding and assumptions that can be made when using dimensional models (Aspect 6)?	6	Kimball *et al.* (1999:148) explain that the dimensional model is gracefully extensible.
14	Discuss what type (if any) monitoring, logging, administration, security auditing or usage logging that occurs on the DW/ETL system.	Preventive maintenance These types of tasks are prescribed by Kimball and Inmon (Aspect 4).	4	Kimball *et al.* (2008:375) - four components to ETL. Inmon *et al.* (2008: 349) – continuous administration of the data warehouse. Kimball *et al.* (2008:380) - subsystems.
15	Describe the volumes data that you typically work with. Does this pose a challenge?	Preventive Maintenance Dealing with high volumes of data (Aspect 7). Regular preventive maintenance would make this less challenging (Aspect 4).	7 4	Inmon *et al.* (2008:118) - Keeping data which is more current on separate hardware. Inmon *et al.* (2008:55) – different technical sectors.
16	How do you overcome the challenge of maintaining high volumes of data?	Preventive Maintenance Follow up to Q15. Partitioning the data (Aspect 8). Separation of data (Aspect 13).	7 4 8 13	Inmon *et al.* (2008:118) - Keeping data which is more current on separate hardware. Inmon *et al.* (2008:55) – different technical sectors.
17	Have you come across strategies for partitioning data? E.g. temporal data (which requires frequent updating and relatively	Preventive Maintenance Follow up to Q14 and to understand the use of Inmon	4 8 13	Inmon *et al.* (2008:118) – separation of stable and unstable data

	Question	Research area	Aspect	Reference
	static data (which does not require frequent updating).	recommendations on separation of temporal and stable data (Aspect 8, 13).		allowing changing requirements to have minimal impact on the entire system.
18	If you have experience as a consultant, talk about experience of DW dev. / maintenance as being a consultant as opposed to being in-house.	Overall Question Weariness of consultants (Aspect 11).	11	Inmon *et al.* (2008:360) – managing consultants. Kimball *et al.* (2008: 580) – keeping the same DW team for maintenance.
19	Tell us about your experience in OLTP systems development: In terms of use of methodologies and in terms of maintenance How does OLTP differ from DW in terms of development? How does OLTP maintenance differ from DW maintenance?	Overall Question Allow participant to compare OLTP and DW maintenance methodology (Aspect 3). There is the possibility for many, if not all aspects to be considered here.	MULTI 1	Research objectives supported.
20	Do you have experience with Agile methodologies or other iterative systems development methodologies? From your perspective, are there similarities in DW development and Agile projects in terms of methods and maintenance?	Methodology The iterative nature of agile methodologies makes them comparable to DW development strategies and perhaps maintenance issues are comparable (Aspect 3).	1	Overhage *et al.* (2010:3), Highsmith and Cockburn (2001: 121) – communication. The Agile Alliance, (2001) – more flexible to additional requirements.
21	Do you feel that DW development and maintenance is more comparable to sequential development than Agile development e.g. the Waterfall methodology?	Methodology/Overall maintenance Follow up on Q20. Answers to Q20 might not have taken maintenance into consideration. This addresses maintenance specifically.	1	Overhage *et al.* (2010:3), Highsmith and Cockburn (2001: 121) – communication. The Agile Alliance, (2001) – more flexible to additional requirements
22	Are you using any industry standard or framework e.g. ISO standards for maintenance? If so, please describe the maintenance process.	Methodology, Corrective/Adaptive maintenance Standards will assist with program comprehension which is a challenge in corrective/adaptive maintenance.	n/a	Thayer and McGettrick (2007: 229) - standards are a basis for software growth, improved communication and easier inter-changeability of developers between projects.
23	What are from your perspectives the advantages and disadvantages of using consultants in DW projects in terms of maintenance?	Corrective/Adaptive maintenance Literature cautions against use of consultants. Using consultants for maintenance could mean a learning curve is required (Aspect 11).	11	Inmon *et al.* (2008:360) – managing consultants. Kimball *et al.* (2008: 580) – keeping the same DW team for maintenance.

	Question	Research area	Aspect	Reference
24	How do development tools assist in maintenance? Have you used these in transactional maintenance or DW maintenance?	Overall Question Tools have been developed and used by OLTP developers. Has the industry as a whole given such attention to maintenance (Aspect 14)?	14	Veerman (2007: 353) Maintenance automation help alleviate maintenance pressures and errors caused by making manual changes.
25	Do you have experience with documentation tools in transactional or DW maintenance? Can you explain how these work and if you find them useful?	Overall Does the use of tools minimize documentation effort (Aspect 14)? Junior IT professionals may not understand that metadata is useful in documentation creation hence the question on how these tools work (Aspect 10, Aspect 5).	5 14 10	Sulaiman *et al.* (2002: 566) – tool usage for documentation

Table 6-2: Interview questions, aims and reference

The interpretive questionnaire was quite lengthy and some themes were repeated to cater for the possibility that the participant may be able to provide input on a topic if the question was approached from a different perspective. For example, Q25 deals with documentation tools which indirectly pose a question about the use of metadata for documentation. The possibility of using metadata for documentation creation was also covered in Q7.

6.4 Data collection: Process and participants

Once the questions were finalised, participants were recruited and interviews began. Section 6.4.1 summarises the interview process and Section 6.4.2 provides details of the participants involved.

6.4.1 Interview process

During the time of the empirical research, the researcher was located in the United States. This posed many challenges as in-person interviews were not possible. Face to face interviews were only possible through video conferencing tools such as Skype. The time difference also posed a challenge to scheduling. Due to these challenges, the option to answer the interview questions in the form of an interpretive questionnaire was utilised when the participant preferred it.

Many email requests were sent out to potential participants which included the interpretive questionnaire. The introduction to this interpretive questionnaire discussed the purpose of the interpretive questionnaire, the process of the interview and gave assurance of anonymity. It also outlined that the participant could either answer via telephonic interview, video conference interview, send audio responses to the questions or return the interpretive questionnaire with answers written down. The introduction and the interpretive questionnaire are included in Appendix A. The data CD contains the interview transcripts.

There were approximately thirty-five requests sent out and eleven people completed the interpretive questionnaire in order to become participants in the study. Of these, there were five interviews conducted telephonically and 6 participants responded in writing. When performing telephonic interviews, the researcher had more opportunity to engage the participant in conversation, and provide any explanation on a question or rephrase the question if required. The written responses did not allow for this. This resulted in some responses being structured with question numbers and answers correlating to similar themes whilst the other interviews were only semi-structured. The semi-structured interviews did have similar questions; however the question numbers and themes do not always correlate to the question numbers and themes of the other responses.

For example, as indicated in Coding Excerpt 1, participant P2, who was interviewed telephonically, was asked about the challenges of DW maintenance:

Coding Excerpt 1: P2 response to challenges of DW maintenance
Shakeel: Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced and how you deal with handling of these challenges. AB: Challenges experienced where mostly understanding of a tool even with relevant training and applying the tool to different environments. Shakeel: Can you elaborate on what you mean by different environments.

Coding Excerpt 1: P2 response to challenges of DW maintenance
AB: Linking to different warehouses, understanding the data which is the most important thing and building it according to the standards of a data warehouse: the Kimball standards and the Inmon standards. You try to get the data to be in that form. The challenge comes in when data comes in differently from different source systems. You need to find a way to link this data together coming in from different sources systems and move it to the proper standards of a data warehouse environment.

With the telephonic interview, the opportunity to ask a follow up question led to the participant referring to the DW methodologies of Kimball and Inmon. On the other hand, participant P3 responded in writing. His response to the same question is shown in Coding Excerpt 2.

Coding Excerpt 2: P3 response to challenges of DW maintenance
Shakeel: Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced. DN: QlikView allows for easy updates and changes due to structure. The biggest challenges are related to the impact of existing data and formulas being effected by the implemented change. Secondary to that is client related issues where the scope was not properly documented and communicated.

A follow-up question regarding graceful changes of dimensional models might have been posed if this had been a telephonic interview.

Kaplan and Maxwell (2005b: 40) explain that when using open ended interviews, one can use the opportunity to request elaborations on responses which might yield more applicable responses. As such, these differences between the written and verbal responses can be expected. However, effort was made by the researcher to make the written interpretive questionnaire understandable and comprehensive and effort was also made during verbal interviews to not deviate drastically from the interpretive questionnaire. This had the effect that the written responses were quite verbose and satisfactory and the verbal and written responses were similar enough to each other to be comparable however

comparison and analysis on a per question basis was not very effective. The comparison and analysis is discussed in Section 6.5.

6.4.2 Participant detail

The eleven respondents are listed in Table 6-3 and named P2 to P12 under Atlas Doc Reference. The list begins at P2 as P1 was duplicated during the coding process and was removed. The Atlas document reference is the reference used by the textual analytical software, Atlas.ti. This code shall also serve as the reference for the participant in this Chapter. The document name is the name of the file containing the transcript of the interview for a specific participant. These files are contained on the data CD. Response type indicates whether the response was made in written format or whether the participant was interviewed. As discussed in Section 6.4.1, the interview response allowed questions to be explained or expanded on as well as allowed for follow up questions. Other data such as the participants' experience, qualification and training were collected. This information might give insight into individual's answers and views and might serve as basis for further comparative studies.

Atlas Doc Reference	Document Name	Response Type	Age	Years In IT	Years In Software Dev.	Years In DW	Highest Qualification	DW/Bi Training
P2	AB	telephonic interview	30	8	4	4	N.Dip	SAP/Microsoft
P3	DN	Written	33	11	2	4	M.Bl	SAP/Qlkview
P4	MW	Written	40	21	21	8	none	Microsoft
P5	MV	Written	38	16	7	6	B.Sc	TDWI
P6	OF	Written	46	20	5	15	B.SEE	Microsoft
P7	KS	Written	40	8	6	3	B.Sc	NA
P8	MN	telephonic interview	36	10	10	7	B.Sc Hons	SAP
P9	SM	Written	42	15	12	5	B.Sc	Microsoft
P10	AN	telephonic interview	44	12	12	4	B.Com	Cognos/SAP
P11	GW	telephonic interview	33	9	8	5	none	Microsoft
P12	SR	telephonic interview	38	13	13	6	B.Com	Microsoft/Cognos

Table 6-3 Participants

6.5 Data analysis

In Chapter 2, research methodologies were explored and in Section 2.6.4, it was decided that that content analysis is the most appropriate method for this research. Section 6.5.1 provides an outline of the strategy of analysis that was followed. As part of Section 6.5.1, the modes of analysis are defined and the process of content analysis is revisited.

Hsieh and Shannon (2005: 1279) discuss three types of content analysis. These are conventional, or undirected content analysis, directed content analysis and summative content analysis.

Undirected content analysis is aimed at description as existing theories on a phenomenon may be limited and the phenomenon needs to be explored further. When a phenomenon has already been explored to some detail and theories have been developed on this phenomenon, the goal of the content analysis may be to validate these theories.This is directed content analysis. The third type of content analysis uses counts and summaries of phrases to induce meaning in the text being analysed. This is known as summative content analysis. In this case, undirected content analysis is appropriate as the research describes the accounts of data warehouse developers in a South African context and there is limited existing theories on this. Furthermore, the research describes the accounts of challenges of maintenance in a data warehouse environment and there is also limited existing literature on this.

A literature review had been conducted and existing data warehouse methodologies have been explored to understand existing challenges and understand how these challenges are addressed within the methodologies. Based on this knowledge an interpretive questionnaire was developed to support the theoretical knowledge empirically. There is a lack in existing literature of empirical evidence of maintenance challenges in data warehousing environments and there was no literature found which pertains to a South African context. Although research areas

were identified while developing the interpretive questionnaire, undirected content analysis was performed to get a clear understanding of the perspectives of the participants. Section 6.5.2 reports on the undirected coding process.

Consolidation and grouping of codes was required after coding. This entailed deduplication of codes and linking together of similar codes. This process is detailed in Section 6.5.4.

6.5.1 Strategy

Myers (1997: 241) discusses three modes of textual analysis: hermeneutics, semiotics and narrative and metaphor. Hermeneutics is concerned with the meaning gained from interpretation of text. When a piece of text is divided into multiple parts, then, as Myers (1997: 241) explains the hermeneutic circle refers to gaining more understanding of the entire text by analysing some of its parts, then applying the understanding to the whole text and then analysing more of its parts and applying that understanding to the whole text. This circle continues until the text is fully analysed.

Semiotics refers to discovering pattern in text or a situation. Myers (1997: 241) explains that the researcher then analysis the frequencies of the patterns and makes deductions based on this.

Narrative and metaphor has its origins in literary studies and have recently become more widespread in other disciplines (Myers, 1997: 241). The narrative mode of analysis involves gaining understanding through a story being related and a metaphor is gaining understanding through comparison of another object.

In this study, we are performing textual analysis which is cyclical by nature. The entire body of text that is being analysed is all the transcribed interviews. A part of this could be a single interview, a question, or even an idea expressed in part of the

response to the question. Through analysis of parts of the entire body of text, more understanding is gained of whole text. This mode of analysis is therefore hermeneutics.

As discussed in Chapter 2, Section 2.6.2.3, Zhang and Wildemuth (2009: 3) promote an eight step process to performing content analysis. Table 6-4 below shows how each of these steps have been applied in this study.

	Step name	Application in this research
1	Prepare the data.	This was performed by transcribing of interviews and removing personal and company identifying data as described in Section 6.5.2.
2	Define the unit of analysis.	Undirected content analysis is being performed and the research seeks out the experiences and challenges of DW professionals. These challenges and experiences were documented and each of these documents represents a unit of analysis as discussed in Section 6.5.3.
3	Develop categories and a coding scheme.	Research areas and aspects for investigation were identified in literature. These were used to develop the interpretive questionnaire. The main categorisation of data is done after undirected coding of Step 5. This process is described in Section 6.5.4.
4	Test your coding scheme on a sample of text.	Three interviews were coded as the initial sample..In Section 6.5.5, some excerpts from these are presented with the discussion of how they were tested.
5	Code all the text.	The rest of the interviews were coded and described in Section 6.5.6 and as new codes were created, they were noted as being new along with the corresponding interview in which the code was first created. These notes were used in the next step of coding where consistency was verified.
6	Assess the coding consistency.	Using the notes created when new codes were created, previously coded interviews were reassessed for consistency with these new codes. Related codes were grouped together to ensure consistency in reporting as discussed in Section 6.5.7.
7	Draw conclusions from the coded data.	After using Atlas.ti to create code categories, the codes grouped together where appropriate to allow for more focused analysis which is presented in Section 6.6.
8	Report your methods and findings.	Chapter 2, 6 and 7 serve as the report of the methods and findings.

Table 6-4 Zhang and Wildemuth's (2009: 3) 8 step process to performing content analysis

Table 6-4 will be referred throughout the analysis report to guide the reader. Steps 1 to 6 of Table 6-4 are discussed under this section of Data Analysis. Step 7 is discussed in Section 6.6 under the heading of "Conclusions from the data".

6.5.2 Step 1 – Prepare the data

Step 1 of Table 6-4 was done as reported in the table and discussed in Section 6.4 above. Interviews that were conducted telephonically were transcribed. The responses that were returned in written format were edited to ensure that no identifiable data e.g. name of employer, was included in the response. The written responses were not corrected in terms of spelling, grammar and punctuation. Any colloquialisms used during the verbal interview process were included in the transcription to demonstrate the level of comfort that the participants may have felt.

6.5.3 Step 2 – Define the unit of analysis

Defining a unit of analysis was the next step of the research process and corresponds to step 2 of Table 6-4. When performing summative content analysis, a unit of analysis may be defined to be something that one shall be able to analyse with numbers and statistics, e.g. a word or sentence. With undirected content analysis, the unit of analysis relates to the expression of a concept.

Initially, the unit of analysis was defined as a single question and the response to that question as it was thought that this would be adequate to compare different expressions of a concept between participants. During the course of testing the analysis, which is discussed in Section 6.5.5, it was clear that a better understanding of concepts could be achieved when comparing an entire response to the interpretive questionnaire than comparing each question individually. This process of examining a single question and then returning to an entire response to gain a better understanding corresponds to a hermeneutic cycle.

6.5.4 Step 3 – Develop categories and a coding scheme

In Table 6-1, research areas and aspects to be investigated were identified from Chapter 5. These aspects were applied to develop the interpretive questionnaire as presented in Table 6-2. In a sense, these aspects to be investigated served as categorisation of the questions however, as undirected content analysis was performed, categories could only be developed after the undirected content analysis was completed.

6.5.5 Step 4 – Testing the coding scheme on a sample of text

As we are performing undirected content analysis, this step involves coding the text, with an undirected approach according the chosen unit of analysis. The initial unit of analysis was chosen to be a response to a question.

To test the undirected content analysis coding strategy, three interviews were coded. Coding Excerpt 3 contains the codes that were inputted into the software, Atlas.ti, for P2's and P3's response to the same question regarding the challenges of data warehousing. The comparison of their responses is discussed below.

Coding Excerpt 3: Codes for P2 's and P3's response to challenges of DW maintenance

P2
Codes:
[Challenge of learning new system - Families (2): Challenges experienced, Training] [materialized view problem - Family: Dimensional modelling] [multiple sources challenge - Families (2): Challenges experienced, Dimensional modelling] [Simplicity of Dimensional models - Family: Dimensional modelling]

Shakeel: Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced and how you deal with handling of these challenges.

AB: Challenges experienced where mostly understanding of a tool even with relevant training and applying the tool to different environments.

Shakeel: Can you elaborate on what you mean by different environments.

AB: Linking to different warehouses, understanding the data which is the most important thing and building it according to the standards of a data warehouse: the Kimball standards and the Inmon standards. You try to get the data to be in that form. The challenge comes in when data comes in differently from different source systems. You need to find a way to link this data together coming in from different sources systems and move it to the proper standards of a data warehouse environment.

P3
Codes:
[Documentation availability - Family: Documentation] [materialized view problem - Family: Dimensional modelling] [Metadata not used by developer - Family: Metadata] [Scope creep - Family: Communication] [Visual BI aids the relationship between business and DW team - Families (2): Communication, Use of Tools]

Shakeel: Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced and how you deal with handling of these challenges.

Coding Excerpt 3: Codes for P2 ‘s and P3’s response to challenges of DW maintenance
DN: QlikView allows for easy updates and changes due to structure. The biggest challenges are related to the impact of existing data and formulas being effected by the implemented change. Secondary to that is client related issues where the scope was not properly documented and communicated.

As can be seen, there are three different code categories which are relevant to the responses in this question, namely dimensional modelling, documentation and communication. This example shows that, based on the first two responses, comparing and analysing the data with the unit of analysis being defined as a response to a question will lead to many very narrow code groups and categories when the categories are developed. Using a participant’s entire response as a unit of analysis leads to wider code groups and categories allowing for more logical comparisons and analyses to be conducted.

It became clear that rich information was given by a participant on research aspects not explicitly investigated by a specific question, which supported to decision to view the entire interview of a participant. Furthermore, as many questions were developed to investigate the same aspects, it was often the case that answers to different questions were comparable and hence received the same codes. These answers depended on the experience of the participants in the real. In this way, comparison on a question by question basis may be misleading as a participant’s answer to a particular question may not correspond to the codes that are targeted by that question.

Instead of trying to create a hermeneutic circle using the question as the part of the text, a participant’s entire response was more applicable. A response can list all codes contained in it and each code can be assessed by the number of times it appears in a single response, or in the entire textual body. It is therefore easier to analyse a part of the text and apply the analysis to the interpretation of the entire body of text, to create the hermeneutic circle.

6.5.6 Step 5 – Code all the text

In this phase, codes were created in an undirected fashion based on what was found in the text. Using textual analytic software, Atlas.ti, the first four responses were coded. Codes that were created in the previous document were reused during the coding of the following documents. The summary and counts for the first four interviews are shown in an alphabetical list in Table 6-5. The counts for all four documents per code are shown in the totals column. The same summary of codes for all documents is included in Appendix B.

	P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	TOTALS
Ad hoc additions	1	1	0	0	2
Agile type of methodology	0	1	1	0	2
Challenge of duration for loading data	0	0	1	0	1
Challenge of team communication	0	0	1	0	1
Challenges experienced:	5	1	3	2	11
Communication challenges of outsourcing	4	0	0	0	4
Cyclical development not being practised	2	2	1	1	6
Different developers doing maintenance	1	0	0	0	1
Difficulty of using the technology	1	1	0	0	2
Direct contact between business and team	0	0	2	0	2
Documentation availability	0	1	0	0	1
Documentation availability negative	1	1	1	0	3
Documentation availability positive	1	0	0	0	1
Documentation given lower priority	0	1	1	0	2
Documentation tool not useful	0	0	1	0	1
Documentation tool used	0	0	1	0	1
Documentation tools not being used	0	1	0	3	4
Documentation usefulness negative	0	0	2	0	2
Documentation usefulness positive	1	0	0	0	1
Does not make use of Kimball's bus matrix when planning changes	3	1	3	3	10
Does not use dimensional models	0	0	1	0	1
Effectiveness of developer training	1	0	0	0	1
Hardware improvement to cope with volume	0	0	1	0	1
High cost of outsourcing	0	1	1	0	2

	P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	TOTALS
High volumes pose challenge	0	1	2	2	5
High volumes require better design	0	1	0	2	3
Inmon type of philosophy	0	0	1	0	1
Kimball methodology	1	0	0	1	2
Lack of communication between developer and user	2	1	0	2	5
Lack of input from client	1	0	0	0	1
Lack of integration in team	1	0	0	0	1
Lack of preventive/perfective maintenance	1	2	1	0	4
Lengthy documentation is counter productive	0	0	1	0	1
Low priority of perfective maintenance	1	0	0	0	1
Maintenance tools not being used	0	1	1	1	3
Materialized view problem	1	0	1	1	3
Metadata used for documentation	0	0	1	0	1
Metadata not understood by developer	1	0	0	0	1
Metadata not used by developer	1	1	1	1	4
Metadata not used for documentation	0	1	0	2	3
Metadata use is manual	1	0	0	0	1
Methodology linked to technology	0	1	0	0	1
Multiple sources challenge	1	0	0	1	2
Not answered	0	0	3	5	8
Not relevant	0	4	0	0	4
Outsourced developers do more harm by focusing on getting the job done quickly	1	0	0	0	1
Outsourced developers have to spend time learning	1	0	1	1	3
Outsourcing negative	1	1	0	1	3
Outsourcing pose risk of losing knowledge of a system	0	1	0	0	1
Outsourcing positive	0	0	2	0	2
Perfective maintenance is performed	0	0	1	2	3
Preventive maintenance monitoring occurs	0	0	1	0	1
Scope creep	0	1	0	0	1
Simplicity of Dimensional models	0	0	1	0	1
Slowly changing dimensions awareness	0	0	0	2	2
Standards not being used	0	1	1	1	3
Steering committee not being utilised	1	1	1	1	4

	P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	TOTALS
Support activity is granting of access to users	0	1	0	0	1
Team knowledge leveraged	0	0	0	1	1
Technology reliance and dependency	1	2	0	0	3
Technology used to improve performance	1	0	0	1	2
Unaware of Inmon's partitioning strategies	2	1	1	1	5
Visual BI aids the relationship between business and DW team	0	1	0	0	1
Totals:	41	35	42	38	156

Table 6-5 Undirected code count for first four responses

When testing this undirected approach, using a participant's entire response as the unit of analysis, it was found that responses to different questions could be easily compared when they contained the same code. For example, in Code Excerpt 4 below, P2, who responded via telephonic interview, was asked a follow up question which led to the code of "cyclical development not being practised" being inserted. P3, P4 and P5 responded via written interpretive questionnaire either responded to question 2 or question 4 which both led to the code of "cyclical development not being practised". Hence, it was possible to make comparisons and draw similarities from different parts of the text when using the participant's entire response as the unit of analysis.

Coding Excerpt 4: Differences between telephonic interview and written interpretive questionnaire for code "cyclical development not being practised"

Code:
[cyclical development not being practised]

P 2: AB.docx - 2:3 [Shakeel: What about on-going re..] (6:8) (Super)

Shakeel: What about on-going requests and prioritization of user requests

AB: Depends on what the business wants. Prioritization is not based on a first come first serve basis. It is based on how urgent the need is.

P 2: AB.docx - 2:23 [Shakeel: What about making add..] (39:49) (Super)

Coding Excerpt 4: Differences between telephonic interview and written interpretive questionnaire for code "cyclical development not being practised"

Shakeel: What about making additions to a data warehouse?

AB: So, this entails adding fields to a fact table or adding dimensions? Mostly done in development area thereafter moved to a testing environment.

Shakeel: So what is the process if this is done in the development area:

AB: Check what is required and impact on the data warehouse side and problems that can be caused. Create documentation – functional and technical specifications and get the signed off. Do in dev environment. Do dev testing. Then move to testing environment where testing is done by a separate testing company and testing environment. They check for data integrity , data load times, duplicate testing. Thereafter, put into a UAT environment for users to test. Thereafter, moved to production during a scheduled time

Shakeel: Is there a difference between this maintenance process and a new data warehouse process.

AB: Not much of a difference.

P 3: DN.docx - 3:3 [4. How do you deal with handli..] (11:12) (Super)

4. How do you deal with handling and prioritization of on-going requests

On-going requests are prioritised based on resource availability, further prioritisation is left to the client to indicate which request should be implemented first.

P 3: DN.docx - 3:15 [2. In particular, describe the..] (6:7) (Super)

2. In particular, describe the process of making additions to the data warehouse after initial deployment.

When a change is requested by a client that involves a data addition, the requirements are captured by the QlikView developer in conjunction with the client and data expert. These are then scoped and costed, where after the approved changes are implemented.

P 4: MW.docx - 4:2 [2 In particular, describe the ..] (16:24) (Super)

2 In particular, describe the process of making additions to the data warehouse after initial deployment.

We have a 3-phase model:

1. BI developers have a sandbox environment in which they are able to experiment freely in their path to producing an end solution.

2. All proposed changes are captured using SQL Server Data Tools, such that the "source code" is separated from business data, and can be easily reviewed by our DBA, as well as maintained in a version control system (Team Foundation Server).

Coding Excerpt 4: Differences between telephonic interview and written interpretive questionnaire for code "cyclical development not being practised"
3. Accepted changes are entered into the production system by the DBA. **P 5: MV.docx - 5:4 [4. How do you deal with handli..] (10:11) (Super)** 4. How do you deal with handling and prioritization of on-going requests Negotiate with sources (requesters) on deliverable dates. Elevate the decision to management.

Having re-defined the unit of analysis and tested that this will be appropriate for analysis, step 5 in Table 6-4 continued and all the text was coded.

6.5.7 Step 6 – Assess the coding consistency

The sixth step is to assess the coding consistency. Apart from rechecking the codes for consistency, this step also ensured that codes created later in the coding process were included in responses that were coded at an earlier stage. Table 6-6 below contains all codes which were newly created in documents P5 to P12.

P5_MV.docx	Kimball methodology
	Slowly changing dimensions awareness
	Multiple sources challenge (name change from multiple sources)
	Steering committee not being utilised
	Technology used to improve performance
P6_OF.docx	Waterfall type of methodology
	Challenge of data integrity
	Team knowledge leveraged
	Challenge of resource and time constraint
	Metadata used by developer
	Not familiar with Dimensional models
P7_KS.docx	Does not use dimensional models
	Tools used for error handling
	Challenge of outdated documentation
	Low priority of perfective maintenance

	Challenge of high volume addressed through perfective maintenance
	Availability higher priority with OLTP
	OLTP has simpler data retrieval
	Performance more important in OLTP
	OLTP less challenging as reloads not required
	OLTP less challenging as data analysis not required
	OLTP maintenance focus is on system architecture (hardware)
	OLTP maintenance focuses on release management
	High cost of outsourcing
P8_MN.docx	Priority based on deadlines
	Challenge of learning new system
	Learning occurs under pressure
	Ticket driven support process leads to communication challenge
	Challenge of duration for loading data
	Challenge of DW methodology knowledge
	Outsourced developers do more harm by focusing on getting the job done quickly
P9_SM.docx	Priority on understanding business rules
	Users involved in development
	Simplicity of Dimensional models
P10_AN.docx	Adaptive maintenance treated as project
	Users need re-training as DW system not used regularly
	User Interface challenges
	Dedicated user support staff
	User training occurs
	High volume require better user communication
	Maintenance requires experienced professionals
	OLTP forces users to change
P11_GW.docx	DW used to ease the load on transactional system
	Challenge of changing user mindset to use DW
	Documentation tool not useful
	Management does not prioritise perfective maintenance
	OLTP given higher priority for maintenance
	OLTP dev require more testing
	Outsourcing pose risk of losing knowledge of a system
	Development tools not well suited for maintenance

P12_SR.docx	Lengthy documentation is counter productive
	More focus placed on perfective maintenance with OLTP

Table 6-6 Codes that were newly created per document

As documents P5 to P12 were coded, the codes created in these documents were noted manually. Thereafter, the interpretive questionnaire responses which were coded prior to the new codes being created were reassessed. For example, after P5 was coded, P1 to P4 were reassessed with the codes created in P5. In this way, consistency with the new codes and previous documents were ensured.

Code inconsistencies can also occur where there are multiple codes that refer to a similar idea. These are eliminated as codes are grouped together. Atlas.ti allows for the creation of code families. These allow codes to be grouped into similar categories. Code families were useful when filtering select a subset of related codes to work with and analyse.

The codes on the left of Figure 6-1 show the families of codes and the codes on the right show the names of codes which were associated with those family names. The numbers within parentheses, "{}", on the left indicates the number of codes that belong to that family. For example, the code family Communication contains 19 codes. On the right hand pane in Figure 6-1 are 2 columns: grounded and density. These indicate the frequency of the code in all documents and the number of links that the code has to other codes.

Whilst creation of code families were useful to group and filter the codes, reports were most meaningful in Atlas.ti when codes were linked to other codes. Codes were created as a parent code or grouping code for other codes to belong to. These codes were not used as part of the coding within the document, but were used solely for the purpose of having other codes belong to them. In Figure 6-1, the codes "Business communication with Team", "Challenges Experienced", "Developer Training" and "Dimensional Models" are examples of parent codes. It can be noted that these codes all have a value of 0 for Grounded indicating that they do not occur

as codes in the document set. The value of in the column Density indicates the number of links to other codes. It can also be noted that the parent codes have a high density as this was the purpose of their creation. The icons for the codes “Business communication with Team” and “communication challenges of outsourcing” are slightly different than the others and the codes themselves are suffixed with a “~” character. This suffix and icon represents that a comment has been made on those codes during the coding process. The comments made were for the researcher’s personal reference and are irrelevant to the research discussion.

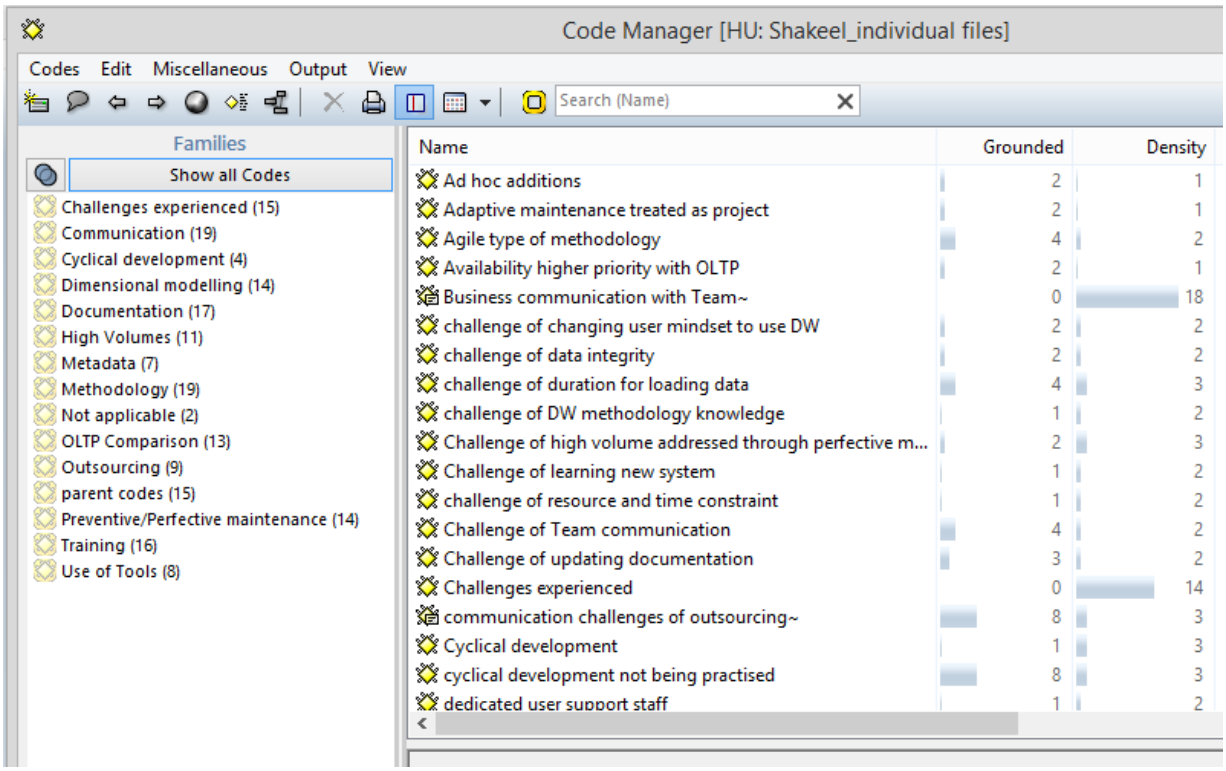


Figure 6 - 1 Code hierarchy from Atlas.ti

Appendix C contains all the codes and the links between them. This is known as a Code neighbour list in Atlas.ti. These relationships demonstrate which codes are recurring and what how they are related. The numbers within the parentheses, “{}”, indicate how many times the code occurs in the entire document set and how many links to other codes this code has. In Code Excerpt 5 below, “Code: cyclical development not being practised {8-3}” shows that this code has occurs 8 times in the document set and has 3 links to other codes. The links to other codes and the relationship between these codes are then listed below, noting that this code is part

of Cyclical Development, Dimensional Models and Methodology Comparison. This example is repeated in Appendix C.

Code Excerpt 5 - example of code neighbour list

Code: cyclical development not being practised {8-3}
<is part of> Cyclical development
<is part of> Dimensional models
<is part of> Methodology comparison

Code: Developer Training {0-6}
Challenge of learning new system <is part of>
Difficulty of using the technology <is part of>
Effectiveness of Developer training <is part of>
Learning occurs under pressure <is part of>
maintenance requires experienced professionals <is part of>
Priority on understanding business rules <is part of>

More detailed reports containing the actual text that was coded along with the codes that are associated to that text were also generated and are included on the CD in rich text format. Each of these reports, correspond to a code family. The file names are prefixed with "Family_" e.g. Family_Methodology.rtf. An excerpt from this file is shown below in Code Excerpt 6.

In the first portion of Code Excerpt 6, prior to the ellipses shows the document heading and a summary of all the codes in the document. Appendix D contains a similar summary of all codes per family. The second part Code Excerpt 6 shows the codes for a particular response to a question.

Code Excerpt 6 – Extract from "All codes with Quotations report"

Code Family: Methodology

Created: 2013-11-24 22:13:45 (Super)
Codes (19): [*Ad hoc* additions] [Adaptive maintenance treated as project] [Agile type of methodology] [cyclical development not being practised] [dedicated user support staff] [Different developers doing maintenance] [DW used to ease the load on transactional system] [Inmon type of philosophy] [Kimball methodology] [Lack of integration in team] [maintenance requires experienced professionals] [Methodology comparison] [Methodology linked to technology] [priority based on deadlines] [Standards not being used] [Steering committee not being utilised] [Technology reliance and dependency] [users involved in development] [waterfall type of methodology]
Quotation(s): 52

…

Code Excerpt 6 – Extract from "All codes with Quotations report"

P 4: MW.docx - 4:2 [2 In particular, describe the ..] (16:24) (Super)
Codes: [cyclical development not being practised - Families (3): Cyclical development, Dimensional modelling, Methodology]
No memos

2 In particular, describe the process of making additions to the data warehouse after initial deployment.

We have a 3-phase model:

1. BI developers have a sandbox environment in which they are able to experiment freely in their path to producing an end solution.

2. All proposed changes are captured using SQL Server Data Tools, such that the "source code" is separated from business data, and can be easily reviewed by our DBA, as well as maintained in a version control system (Team Foundation Server).

3. Accepted changes are entered into the production system by the DBA.

Pieces of text which were not relevant were coded as "not relevant" and questions that were not answered were coded as "unanswered". This was to ensure that no text was left out of the analysis. The reports on these codes are also included on the data CD.

6.6 Step 7 - Conclusions from the data

Apart from reporting of findings in step 8, the last step in Table 6-4 involves analysing the data and drawing conclusions. This step commences in the section and step 8 is handled in Chapter 7.

The analysis on a per code family basis as shown in Figure 6-1 worked well as families represent the themes that emanated from the data. Using the full quotation reports on code families contained in Appendix D, code family summary reports and code counts per document, there was a considerable amount of information available for analysis.

Each of the sub-sections of this section discusses the codes created within each family. The number of codes per document per family initiates this discussion followed by the discussion on the prominently featured codes in this family.

Conclusions are drawn per code family in subsections after the discussion on the code family. These conclusions represent the information divulged from the analysis of the data which in turn represents the perception of maintenance of data warehouses by industry professionals.

6.6.1 Theme 1: Challenges experienced

The challenges experienced by the participants were each coded and a code family was created which contains all the codes that deal with challenges experienced. The count of the occurrences of each of these codes appears in Table 6-7. The most common challenges experienced relates to high volumes of data associated with data warehousing. This challenge also relates to the challenge of data load duration and this is evident as that code also has a higher than average frequency of 4.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Challenge of changing user mind-set to use DW	0	0	0	0	0	0	0	0	0	2	0	2
Challenge of data integrity	0	0	0	0	1	0	0	1	0	0	0	2
Challenge of duration for loading data	0	0	1	0	0	0	2	1	0	0	0	4
Challenge of DW methodology knowledge	0	0	0	0	0	0	1	0	0	0	0	1
Challenge of high volume addressed through perfective maintenance	0	0	0	0	0	1	1	0	0	0	0	2
Challenge of learning new system	0	0	0	0	0	0	1	0	0	0	0	1
Challenge of resource and time constraint	0	0	0	0	1	0	0	0	0	0	0	1
Challenge of Team communication	0	0	1	0	0	0	2	0	1	0	0	4
Challenge of updating documentation	0	0	0	0	0	1	2	0	0	0	0	3
Communication challenges of outsourcing	4	0	0	0	0	2	1	0	1	0	0	8
High volumes pose challenge	0	1	2	2	1	2	3	2	1	1	0	15
Multiple sources challenge	1	0	0	1	0	0	0	0	1	0	0	3

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Ticket driven support process leads to communication challenge	0	0	0	0	0	0	2	0	0	0	0	2
User Interface challenges	0	0	0	0	0	0	0	0	2	0	0	2
Totals:	5	1	4	3	3	6	15	4	6	3	0	50

Table 6-7 Count of codes per document for Family Challenges Experienced

As seen in the Table 6-9 the code "High volumes pose challenge" has the highest total of15. This is because Question 16 deals directly with overcoming high volumes of data hence a high number of responses can be expected. Related to the challenge of high volumes of data is the challenge of data load duration. However, many respondents did not have adequate means of dealing with high volumes. Some examples of this are shown in Code Excerpt 7. In the situation of P4, more disk space is added however this is an unsatisfactory solution for P4. In the case of P11, the challenge of high volumes ought to be addressed by spending more time on overall design, however, this is not always possible.

Code Excerpt 7 – High volumes pose challenge

P 4: MW.docx - 4:12 [16 How do you overcome the cha..] (129:132) (Super)
Codes: [Hardware improvement to cope with volume - Families (2): High Volumes, Preventive/Perfective maintenance] [High volumes pose challenge - Families (2): Challenges experienced, High Volumes]
No memos

16 How do you overcome the challenge of maintaining high volumes of data

At <1TB per database instance, we have haphazardly keeping monthly on-line archives on our SAN.

In terms of moving volumes of data… as per answer 15, we don't really deal with it [well] – it's a problem for us.

P11: GW.docx - 11:17 [16. How do you overcome the ch..] (61:62) (Super)
Codes: [High volumes pose challenge - Families (2): Challenges experienced, High Volumes] [High volumes require better design - Family: High Volumes]
No memos

16. How do you overcome the challenge of maintaining high volumes of data

It's a time consuming, laborious task. It is something that needs to be planned in context of the wider system. When you need to update info in the WH – sometimes other production processes are unable to complete because it waits for this process to finish.

The other challenge that seems to be prevalent amongst data warehousing professionals is that of communication, and in particular, communication when outsourcing is involved. "Challenge of Team communication" was coded 4 times and "Communication challenges of outsourcing" was coded 8 times.

Code Excerpt 8 below contains an excerpt from P8 who works for a multinational company. This company has outsourced maintenance to a company in India. While this might seem like an effective cost saving measure, we note that more affordable outsourcing contracts can have disadvantages. In this case, the lack of experience of the outsourced contractors makes communication difficult with users and co-workers alike. This requires extra time from employees and could be more costly to the company.

Code Excerpt 8– Communication challenges of outsourcing

P 8: MN.docx - 8:28 [23. From your perspective, wha..] (176:179) (Super)
Codes: [communication challenges of outsourcing - Families (3): Challenges experienced, Communication, Outsourcing] [Outsourced developers have to spend time learning - Family: Outsourcing] [Outsourcing negative - Family: Outsourcing]
No memos

23. From your perspective, what are the advantages and disadvantages of using consultants in DW/BI projects in terms of maintenance?
Most companies will use an Indian based consulting house for maintenance. The consultants are normally not very experienced, but with a few that is very good (like 1 person). It is hard to do KT and then expect them to support without being in same vicinity as functional teams and users. Advantage – they keep the small issues away from me and waste their time on their own analysis. Sometimes this is sufficient, but when not I will have to do my own analysis as I rarely trust theirs. Disadvantage – it wastes time to explain to them and investigate. If an issue comes to me I don't want to work through them, I will rather solve it myself. When they mess up it is proper and then I have to step in at the end of the process when people are agitated and the items is months overdue. I used to advise them on what to do, but now I do it myself after one issue last year just due to communication lack – they missed a 'not' in my sentence e.g. do not delete the previous initialise…

P10: AN.docx - 10:23 [23. From your perspective, wha..] (129:133) (Super)
Codes: [communication challenges of outsourcing - Families (3): Challenges experienced, Communication, Outsourcing] [Outsourced developers have to spend time learning - Family: Outsourcing] [Outsourcing negative - Family: Outsourcing]
No memos

23. From your perspective, what are the advantages and disadvantages of using consultants in DW/BI projects in terms of maintenance

Two specific situations come to mind:

They can get very high level to almost be useless. In two of the 3 cases, we got rid of them before too long. In the first one, I didn't see what they were telling us that we didn't already know and then the other case, we spend a lot of time doing interviews with people and the document that was produced was shockingly

Code Excerpt 8– Communication challenges of outsourcing
useless. The person was not taking any notes – and I was thinking that they must be pretty amazing to remember all this. And the reason that they weren't taking any notes was to see if it fits into some high level model and they produced a few pieces of paper that was not helpful to us. So they were supposed to be helping with documentation and process but it was way too high level. In another situation, they were trying draw conclusions and assess the situation but never really understanding it enough to give helpful conclusions. I also think that when people are given a very silo type of job, and you ask them to do something else, they can do that but whether or not that's useful is really questionable so if somebody is going to be given a task, they need to be able to check with other people and not just forge ahead with their job and find out too late that it's not was being sort after. There's not enough communication with this type of role until its way too far down the road. In general, consultants are used to getting a very narrow directive.

P10 feels that outsourced individuals seem to be better at tasks which are narrow. DW tasks tend to be broader, and incorporate larger areas of an enterprise. Hence, DW professionals require a broader understanding of the software system and the business rules that govern the system. This type of understanding is usually cultivated after spending months or even years working for a company and having dealt with multiple facets of it. Outsourced individuals are usually focussed on short term projects and do not spend time gaining a broader understanding of their clients systems. This can be problematic with data warehouses.

6.6.1.1 Conclusions drawn on Challenges Experienced

The greatest challenges seem to be centred on perfective/preventive maintenance and communication. Dealing with higher volumes of data associated with data warehouses and communicating with outsourced individuals are particularly challenging. The challenge of high volumes is one that many professionals struggle with. This in general relates to perfective and preventive maintenance.

Communication challenges are prevalent in DW projects. This can be expected as a DW project includes members from most areas of the company. Scheduling challenges and prioritisation challenges and challenges are not uncommon. When outsourced companies are involved, these challenges can be greater unless the contracts are precise.

6.6.2 Theme 2: Communication

From the previous section on challenges experienced, we know that communication is one of the foremost challenges. Hence, there were many codes dealing with communication. Table 6-8 lists the count of codes per document for the code family "Communication". There are 19 codes in this family which makes it the biggest along with the code family of "methodology".

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Business communication with Team	0	0	0	0	0	0	0	0	0	0	0	0
Challenge of resource and time constraint	0	0	0	0	1	0	0	0	0	0	0	1
Challenge of Team communication	0	0	1	0	0	0	2	0	1	0	0	4
Communication	7	4	4	4	6	4	10	3	5	4	4	55
Communication challenges of outsourcing	4	0	0	0	0	2	1	0	1	0	0	8
Dedicated user support staff	0	0	0	0	0	0	0	0	1	0	0	1
Direct contact between business and team	0	0	2	0	2	0	0	0	0	0	0	4
High volume require better user communication	0	0	0	0	0	0	1	0	1	0	1	3
Lack of communication between Developer and User	2	1	0	2	0	2	3	0	1	1	1	13
Lack of input from client	1	0	0	0	0	0	0	0	0	0	0	1
Lack of integration in team	1	0	0	0	0	0	0	0	0	0	0	1
Management does not prioritise perfective maintenance	0	0	0	0	0	0	0	0	0	1	0	1
Outsourced developers do more harm by focusing on getting the job done quickly	1	0	0	0	0	0	1	0	0	0	0	2
Priority based on deadlines	0	0	0	0	0	0	1	0	0	0	0	1
Priority on understanding business rules	0	0	0	0	0	0	0	2	0	0	0	2
Scope creep	0	1	0	0	0	0	0	0	0	0	0	1
Steering committee not being utilised	1	1	1	1	2	1	2	1	1	2	1	14
Team knowledge leveraged	0	0	0	1	1	0	0	0	0	0	0	2
Ticket driven support process leads to communication challenge	0	0	0	0	0	0	2	0	0	0	0	2
Visual BI aids the relationship between business and DW team	0	1	0	0	0	0	0	0	0	0	1	2
Totals:	17	8	8	8	12	9	23	6	11	8	8	118

Table 6-8 Count of codes per document for Family Communication

Some of the codes, such as “Communication challenges of outsourcing” and “Challenge of team communication” have already been discussed in Section 6.61. The code of “Communication” which was coded 55 times was used as a broad code for anything related to communication. A significant number of entries exist for the codes “Lack of communication between developer and user” and “Steering committee not being utilised”.

Code excerpt 9 contains an excerpt from P12, who is a business analyst and does a lot of user support. The typical DW project, as described by him is that developers do not interact with the users directly. This leaves the developers detached from the problems experienced by the users. This seems to be a common scenario in a data warehousing scenario. P3 describes that the product owner, in an agile environment hands over new requirements. P5 responded that user interaction with developers is not applicable.

Code Excerpt 9 – Communication challenges of outsourcing

P12: SR.docx - 12:10 [8. Describe the relationship t..] (37:40) (Super)
Codes: [lack of communication between Developer and User - Families (2): Communication, Training]
No memos

8. Describe the relationship that DW/BI team has with business users and stakeholders and the interaction that goes on between the DW/BI team and the business users and stakeholders

So in a typical DW project you will have people who interact with users on a regular basis: Customer support or data mining folks or key people who lead the project on a management level but most of the technical or deployment or development team will not interact with the user very much. That’s a typical dw scenario. What I’ll recommend based on our experience is having people from testing and development also meet with the users from time to time to get sense of what the project is all about from a users point of view. This way downstream design can be better done to meet their needs.

Shakeel: So how is user training addressed?

So user training is scheduled based on requests from users and our customer support people going and training the users and then making sure their needs are met based on what the DW offers.

P 3: DN.docx - 3:7 [9. How are the end users suppo..] (23:24) (Super)
Codes: [lack of communication between Developer and User - Families (2): Communication, Training]
No memos

9. How are the end users supported?

From a technical point of view, the clients existing technical process is leveraged as a Technical Resource is made responsible for the QlikView environment. For business however, a product owner is used as contact for new requests – this is then filtered and handed over to the relevant QlikView developer.

Code Excerpt 9 – Communication challenges of outsourcing

P 5: MV.docx - 5:8 [8. Describe the relationship t..] (20:21) (Super)
Codes: [lack of communication between Developer and User - Families (2): Communication, Training]
No memos

8. Describe the relationship that DW/BI team has with business users and stakeholders and the interaction that goes on between the DW/BI team and the business users and stakeholders

N/A

In Code Excerpt 10, some text was coded with "steering committee not being utilised". The code was used when trying to understand the process of how new requests are prioritised as shown on the list of questions in Table 6-2. Kimball *et al.* (1998: 728) emphasises the use of a steering committee to drive the process of growing the data warehouse in a manner which benefits the overall goals of an enterprise. From the responses received, it seems that a steering committee is rarely being used.

Code Excerpt 10 – Steering committee not being utilised

P 9: SM.docx - 9:4 [4. How do you deal with handli..] (24:25) (Super)
Codes: [Steering committee not being utilised - Families (2): Communication, Methodology]
No memos

4. How do you deal with handling and prioritization of on-going requests

Requests are prioritized by the a project manager based on the impact to business.

P11: GW.docx - 11:2 [2. In particular, describe the..] (10:11) (Super)
Codes: [DW used to ease the load on transactional system - Families (2): Methodology, OLTP Comparison] [Steering committee not being utilised - Families (2): Communication, Methodology]
No memos

2. In particular, describe the process of making additions to the data warehouse after initial deployment.

GW: The 2 parts are the user perspective, the high end users who will actually interact with the WH structures – they tell us about the fields needed. Then part 2 when we come into contact with *ad hoc* reports from the transactional system – we then look at moving those to the WH.

P11: GW.docx - 11:4 [4. How do you deal with handli..] (17:18) (Super)
Codes: [DW used to ease the load on transactional system - Families (2): Methodology, OLTP Comparison] [Steering committee not being utilised - Families (2): Communication, Methodology]
No memos

Code Excerpt 10 – Steering committee not being utilised
4. How do you deal with handling and prioritization of on-going requests There's 2 sides to it. Ultimately, if there's a major business need, then the business users decide on. Then, there's the other side which is choosing a report to slot in to the WH that makes IT sense. So, this would be transactional reports which may perform better on the WH.

6.6.2.1 Conclusions drawn on Communication

Kimball *et al.* (2008:136) summarises the roles and level of involvement of those roles in user requirements collection. The data modellers, database administrators and application developers should be involved with user interviews. This encourages an understanding of needs from a user perspective. This is often not the case which leads to communication and support challenges after the project has been deployed; when maintenance begins.

A steering committee representing the entire organisation. This helps build the data warehouse into the single source of reporting for the whole organisation. When one is not being used, data silos are built and different departments use different versions of data for reporting leading to discrepancies and questions around data integrity. This communication challenge is hence also related to an overall methodology challenge. Furthermore, adaptive maintenance in the form of creating additions to the warehouse is hindered because there are challenges around prioritisation of requests. When too many requests are trying to be fulfilled, resources are spread too thin leading to a product being deployed with imperfections. This leads to further issues in support after the product is deployed.

6.6.3 Theme 3: Cyclical development

Both Inmon and Kimball recommend a cyclical approach to data warehouse development in which phases which tend to overlap. Transactional systems development has traditionally followed a sequential approach with distinct phases which each have a start and end point. In order to analyse the different types of methodologies being followed, or not being followed the text was coded where

cyclical development was being practised or not being practised. The counts for these codes are shown in Table 6-9. As most agile methodologies are also cyclical, these were included in the code family.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Agile type of methodology	0	1	1	0	0	0	0	1	0	1	0	4
Cyclical development	0	0	0	0	0	0	0	1	0	0	0	1
Cyclical development not being practised	2	2	1	1	0	1	0	0	0	0	1	8
Inmon type of philosophy	0	0	1	0	0	1	0	0	0	0	0	2
Totals:	2	3	3	1	0	2	0	2	0	1	1	15

Table 6-9 Count of codes per document for Cyclical Development

6.6.3.1 Conclusions drawn on Cyclical Development

As can be seen from Table 6-9, P2, P3, P4, P5, P7 and P8 do not practise cyclical development. This is six out of the 12 participants. This is a key finding for this research as it indicates that data warehouse professionals do not follow recommended guidelines for data warehouse development.

6.6.4 Theme 4: Dimensional models

The use of dimensional models in data warehouse development is dependent on the chosen methodology. Practitioners who practice a Kimball approach to data warehousing should be using dimensional models. These practitioners should also make use of the dimension bus architecture created by Kimball *et al.* (1998; 271) as this is a fundamental component of dimensional modelling as it ensures consistency of data and it also ensures that time and resources are effectively spent as work is not duplicated to load the same data into the data warehouse.

Table 6-10 below shows the counts of codes per document that belong to the code family “dimensional modelling”. The highest code count occurs for “Does not make use of Kimball's bus matrix when planning changes”.

	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9	P1 0	P1 1	P1 2	TOTAL S
Ad hoc additions	1	1	0	0	0	0	0	0	0	0	0	2
Challenge of DW methodology knowledge	0	0	0	0	0	0	1	0	0	0	0	1
Cyclical development not being practised	2	2	1	1	0	1	0	0	0	0	1	8
Different developers doing maintenance	1	0	0	0	0	0	1	0	0	0	0	2
Does not make use of Kimball's bus matrix when planning changes	3	1	3	3	2	0	0	1	1	0	0	14
Does not use dimensional models	0	0	1	0	1	1	0	0	1	0	0	4
Kimball methodology	1	0	0	1	0	0	0	1	0	1	0	4
Materialized view problem	1	0	1	1	0	1	0	0	0	0	1	5
Multiple sources challenge	1	0	0	1	0	0	0	0	1	0	0	3
Not familiar with Dimensional models	0	0	0	0	1	0	0	0	0	0	0	1
Simplicity of Dimensional models	0	0	1	0	0	0	0	1	0	2	0	4
Slowly changing dimensions awareness	0	0	0	2	0	0	0	0	0	0	0	2
Slowly Changing Dimensions/graceful changes	0	0	0	0	0	0	0	0	0	0	0	0
Totals:	10	4	7	9	4	3	2	3	3	3	2	50

Table 6-10 Count of codes per document for Dimensional modelling

Amongst the challenges experienced with dimensional models was the issue of materialized views. This is the challenge of keeping a copy of the source data current on the data warehouse. This problem is exacerbated when there are multiple sources of data. Code excerpt 11 below shows that this problem is a difficult one to deal with. P7 suggests that more time needs to be spent understand the source data and its format. P10 speaks of difficulties with dealing with source systems which one does not have control over. When something changes on these systems, it could result in a failure on the data warehouse system.

Code Excerpt 11 – Materialized view problem

P 7: KS.docx - 7:5 [4. How do you address the chal..] (19:25) (Super)
Codes: [materialized view problem - Family: Dimensional modelling] [Technology reliance and dependency - Family: Methodology]
No memos

4. How do you address the challenge of working on a data warehouse that you have not worked

on before?

Understand more about the source Data and its storage format. The growth of data on

the source should be analysed in depth to plan the extraction and cleansing the data from the source and select the appropriate software tools and hardware infrastructure to meet the needs of growing data with respect to extraction and storage of data .

For an example Data warehousing project is a read intensive system so selecting a column oriented database can be considered in addressing the challenging of the performance issues. The BI Tools can be evaluated as well

Taking the list of lack of features in data ware housing tools used in the previous project which resulted in spending more time by the developers to for an workaround otherwise the time and effort could have been reserved for something useful.

P10: AN.docx - 10:3 [3. Talk about some of the chal..] (25:34) (Super)
Codes: [mulitple sources challenge - Families (2): Challenges experienced, Dimensional modelling] [User Interface challenges - Families (2): Challenges experienced, Training]
No memos

3. Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced.

1 comes to mind is that we're relying on various sources to give us the data. The upstream from us. So how data is configured upstream determines how we can pull it out as discrete data and so we don't have a direct a line to say "before you deploy your system, let us do some testing". That's not done.

Shakeel: It's not having any control on the source data.

AB: Right and then there are issues with the data. E.g. data comes over from 1 system as large text or word documents. It may look good as a text document but we are trying to pull the discrete data - the configuration is set up for that so we have problems pulling the data elements discretely. It looks o.k. when it's on 1 big text report. That's happened in a number of areas.

Then you have various area having "ownership" of the data.

Shakeel: What about user challenges and user education challenges?

AB: Um, usability is a big one. The system needs to more intuitive – there's no question about it. The usability side, the intuitiveness.

6.6.4.1 Conclusions drawn on Dimensional modelling

Practitioners of data warehouse seem to be unaware of the Kimball *et al.*'s (1998; 271) bus matrix architecture. During the analysis of the responses, there was no suggestion of any other method to ensure data consistency or prevent duplication of effort. This could lead to various maintenance challenges. As data is duplicated, volumes increase resulting in preventive and perfective maintenance challenges. When different versions of the same data appear in different reports it creates communication challenges and user support challenges. Inefficient use of resources

leads to extra pressure to deliver, resulting in imperfect products being deployed creating corrective maintenance challenges.

6.6.5 Theme 5: Documentation

Documentation is a topic that most DW professionals had a comment on. Hence there are 17 codes created under the documentation family and a total of 65 code occurrences. Q6 and Q7 were directly related to documentation hence there are a large number of codes in the family. Table 6-11 contains the counts per document on the numbers of codes.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Challenge of updating documentation	0	0	0	0	0	1	2	0	0	0	0	3
Difficulty of using the technology	1	1	0	0	0	0	0	0	0	0	0	2
Documentation availability	0	1	0	0	0	0	0	0	0	0	0	1
Documentation availability negative	1	1	1	0	1	0	1	0	0	0	0	5
Documentation availability positive	1	0	0	0	0	0	0	1	1	0	0	3
Documentation given lower priority	0	1	1	0	0	0	0	0	0	0	0	2
Documentation including use of metadata	0	0	0	0	0	0	0	0	0	0	0	0
Documentation tool not useful	0	0	1	0	0	0	0	0	0	2	0	3
Documentation tool used	0	0	1	0	0	0	0	0	0	0	0	1
Documentation tools not being used	0	1	0	3	1	1	2	2	1	0	1	12
Documentation usefulness negative	0	0	2	0	0	1	2	0	0	2	1	8
Documentation usefulness positive	1	0	0	0	1	0	0	1	1	0	0	4
Lengthy documentation is counter productive	0	0	1	0	0	1	0	0	0	0	1	3
Metadata used for documentation	0	0	1	0	0	0	0	0	0	0	0	1
Metadata not used for documentation	0	1	0	2	1	0	1	0	0	0	1	6
Metadata use is manual	1	0	0	0	0	0	0	0	0	0	1	2
Standards not being used	0	1	1	1	0	1	1	1	1	1	1	9
Totals:	5	7	9	6	4	5	9	5	4	5	6	65

Table 6-11 Count of codes per document for Dimensional modelling

Three of these codes are also within the metadata family. These are “documentation tool used”, “Metadata used for documentation”, “Metadata not used for documentation” and “Metadata use is manual”. In general, it was found that while

there should be a relation between metadata and documentation, this is seldom the case.

A surprising finding from the responses was that many participants felt that documentation is not very useful. Code excerpt 12 below contains an extract from P4 and P11's responses. They feel that documentation is not useful or it contains data that can be found easily by looking up the properties of the database object. P11 expands on this by saying that documentation is not useful when it is not updated. This is often the case.

Code Excerpt 12 – Documentation usefulness negative

P 4: MW.docx - 4:4 [5 How do you address the chall..] (43:64) (Super)
Codes: [Documentation availability negative - Family: Documentation] [Documentation usefulness negative - Family: Documentation] [lengthy documentation is counter productive - Family: Documentation]
No memos

5 How do you address the challenge of working on a data warehouse that you have not worked on before?

The biggest challenge here usually a lack of meaningful documentation.

Superb data warehouses can be relatively self-documenting purely on account of the naming conventions followed during construction, but the reality is that most data warehouses are build bottom up which tends to bring much of an OLTP source system's obscurity into the warehouse.

Documentation on inherited data warehouses tends to be either:

1. non-existent, or

2. painfully verbose.

By painfully verbose, I mean that the documentation goes to great lengths to spell out the readily obvious – i.e. hundreds of pages detailing the name and data type of every column in every table in the warehouse. This sort of information is readily available in most development environment tools, and doesn't go very far in describing why certain tables and columns exists.

P11: GW.docx - 11:8 [7. Describe when and how docum..] (24:27) (Super)
Codes: [Documentation tool not useful - Family: Documentation] [Documentation usefulness negative - Family: Documentation]
No memos

7. Describe when and how documentation is created.

Tools kind of suck things out of databases which is not very useful. I think a doc might only be useful if there's some sort of thought that goes into it. Where somebody goes and investigates something and puts down the results of that sort of investigation. However, people need to actively manage that document.

Code Excerpt 12 – Documentation usefulness negative
That's a lot better than a tool which documents a whole database. In practice, somebody would be tasked with documenting the WH and the document gets put in a folder structure somewhere and it rarely gets looked at again. Because of that, you get inconsistences in naming in the naming conventions, etc. Also because people aren't held accountable through the document. The document should be the standard that everyone abides by. Where, it should be mandatory to read and follow. Otherwise, what's the point of the document is there's no accountability- the document should be the agreed upon process.

In Code Excerpt s 11 and 12, thoughts are also expressed by participants regarding the documentation length. Below, in Code Excerpt 13, P12 considers that a lot of effort goes into documentation creation. Unless this document is usable, this effort is wasted.

Code Excerpt 13 – lengthy documentation is counter productive
P12: SR.docx - 12:8 [6. Describe the use of documen..] (26:30) (Super) Codes: [Documentation usefulness negative - Family: Documentation] [lengthy documentation is counterproductive - Family: Documentation] No memos 6. Describe the use of documentation in maintaining and supporting data warehouses. Documents are good in the sense that they enforce structure and lead to more thinking so having documentation is critical but at the same time a balance is needed. Excessive documentation can lead to limitations downstream which will not allow flexibility in design as opposed to there being less documentation. So there has to be a limit on documentation to make it useful. For example if we build a 200 page requirements document, not many developers will read it end-to-end to make sense of it or they'll miss the important parts. Shakeel: Has that been your experience? SR: It has been, yes. It's very important to maintain limits on how long a document should be – as this determines how usable it is and how readable and clear it is and how easy it is to follow – both requirements and user guides and other things. Unless it's usable, there's no point in building or creating those documents.

6.6.5.1 Conclusions drawn on Documentation

While many participants found documentation useful, more did not. The problem may be that documentation is out of date or that the information contained in the document is too lengthy or even irrelevant. This stems from documentation not being

a priority. It was mentioned in Chapter 3 that Selic (2009: 11) warns that poorly documented systems lead greater challenges and costs during maintenance.

Another code which stands out in Table 6-11 is that standards are not being utilised. This could increase communication challenges as people from different backgrounds may have different notions of definitions. For example, one might use the word functional specification to include only an end user specification or one might also include database connectivity definitions in a functional specification. Standards enhance communication and streamline project processes.

6.6.6 Theme 6: High volumes

High volumes of data is one of the most common challenges in data warehouse maintenance. This was discussed in Section 6.6.1. Table 6-12 below lists the other codes that are related to high volumes of data. The more prevalent codes are related to perfective maintenance and Inmon's partitioning strategies. The code "Unaware of Inmon's partitioning strategies" was created for Q17. Most respondents were unaware of the strategy of placing temporal data and static data in separate partitions.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Challenge of duration for loading data	0	0	1	0	0	0	2	1	0	0	0	4
Challenge of high volume addressed through perfective maintenance	0	0	0	0	0	1	1	0	0	0	0	2
Hardware improvement to cope with volume	0	0	1	0	0	0	2	0	0	1	0	4
High volume require better user communication	0	0	0	0	0	0	1	0	1	0	1	3
High volumes pose challenge	0	1	2	2	1	2	3	2	1	1	0	15
High volumes require better design	0	1	0	2	1	0	1	2	0	1	0	8
Lack of preventive/perfective maintenance	1	2	1	0	1	0	3	0	0	0	0	8
Perfective maintenance is performed	0	0	1	2	2	2	0	2	1	0	1	11
Preventive maintenance monitoring occurs	0	0	1	0	0	0	0	0	1	0	1	3
Unaware of Inmon's partitioning strategies	2	1	1	1	1	1	1	1	0	1	0	10

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Volume of data	0	0	0	0	0	0	0	0	0	0	0	0
Totals:	3	5	8	7	6	6	14	8	4	4	3	68

Table 6-12 Count of codes per document for High Volumes

6.6.6.1 Conclusions drawn on High Volumes

It was encouraging to note that perfective maintenance is being performed in many cases. There were, however, also cases where a participant who reported some sort of perfective maintenance being implemented also reported that there is a lack of perfective maintenance. This indicates that perfective maintenance is an on-going task which project managers and operational managers should always consider.

6.6.7 Theme 7: Metadata

Some of the questions targeted the use of metadata. Table 6-13 shows that only 2 participants reported that metadata is being used whereas 6 reported that it is not being used.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Metadata used for documentation	0	0	1	0	0	0	0	0	0	0	0	1
Metadata not understood by developer	1	0	0	0	0	1	0	0	0	0	0	2
Metadata not used by developer	1	1	1	1	0	1	0	0	0	1	0	6
Metadata not used for documentation	0	1	0	2	1	0	1	0	0	0	1	6
Metadata use is manual	1	0	0	0	0	0	0	0	0	0	1	2
Metadata used by developer	0	0	0	0	1	0	0	0	0	0	1	2
TOTALS:	3	2	2	3	2	2	1	0	0	1	3	19

Table 6-13 Count of codes per document for Metadata

6.6.7.1 Conclusions drawn on Metadata

During telephonic interviews, the researcher felt that this topic was not well understood by data warehouse practitioners. While most practitioners know what the definition of metadata is, there is still some uncertainty as to its practical applications. In Code Excerpt 14 below, P7 has some idea of metadata in practice however his response should contain a more complete description.

Code Excerpt 14 – Metadata not understood by developer
P 7: KS.docx - 7:12 [10. Talk about your experience..] (44:45) (Super) Codes: [Metadata not understood by developer - Family: Metadata] [Metadata not used by developer - Family: Metadata] No memos 10. Talk about your experience with the practical application of metadata in the maintenance of data warehouses and the other practical uses of metadata. The data type change in the back end table causes the meta data to synchornize with the change resulting in re-deployment of the packages.

Although the response from P7 was incomplete, he had a better handle of metadata than most other participants. This is evident when one examines Table 6-13 as most participants did not discuss much on the topic of metadata.

6.6.8 Theme 8: Methodology

Methodology is a broad code family. It contains many codes which have already been discussed. These include "Cyclical development not being practised", "Standards not being used" and "Steering committee not being utilised" as shown below in Table 6-14 below.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Ad hoc additions	1	1	0	0	0	0	0	0	0	0	0	2
Adaptive maintenance treated as project	0	0	0	0	0	0	0	1	1	0	0	2
Agile type of methodology	0	1	1	0	0	0	0	1	0	1	0	4
Cyclical development not being practised	2	2	1	1	0	1	0	0	0	0	1	8
Dedicated user support staff	0	0	0	0	0	0	0	0	1	0	0	1
Different developers doing maintenance	1	0	0	0	0	0	1	0	0	0	0	2
DW used to ease the load on transactional system	0	0	0	0	0	0	0	0	0	2	0	2
Inmon type of philosophy	0	0	1	0	0	1	0	0	0	0	0	2
Kimball methodology	1	0	0	1	0	0	0	1	0	1	0	4
Lack of integration in team	1	0	0	0	0	0	0	0	0	0	0	1
Maintenance requires experienced professionals	0	0	0	0	0	0	1	0	1	0	0	2
Methodology linked to technology	0	1	0	0	0	0	0	0	0	0	0	1
Priority based on deadlines	0	0	0	0	0	0	1	0	0	0	0	1
Standards not being used	0	1	1	1	0	1	1	1	1	1	1	9

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Steering committee not being utilised	1	1	1	1	2	1	2	1	1	2	1	14
Technology reliance and dependency	1	2	0	0	0	1	1	0	0	0	0	5
Users involved in development	0	0	0	0	0	0	0	1	0	0	0	1
Waterfall type of methodology	0	0	0	0	1	0	1	0	1	0	1	4
Totals:	8	9	5	4	3	5	8	6	6	7	4	65

Table 6-14 Count of codes per document for Methodology

Two participants agree with the Inmon philosophy of not using dimensional models. There were no mention of Inmon's DW 2.0 methodology in the responses, however, as can be seen in Code Excerpt 15 below, P4 and P7 feel that dimensional modelling has shortfalls.

Code Excerpt 15 – Inmon type of philosophy

P 4: MW.docx - 4:9 [13 Describe the advantages or ..] (112:117) (Super)
Codes: [Does not use dimensional models - Family: Dimensional modelling] [Inmon type of philosophy - Families (3): Cyclical development, Methodology, Slowly changing dimensions]
No memos

13 Describe the advantages or disadvantages of maintaining dimensional models.

Dimensional models tend to attempt to produce a "confirmed representation" of business data that can be easily and consistently aggregated. When done correctly, the obvious advantage is the ability for stakeholders to explore hidden business value in a fluid manner.

In practice, dimensional models tend to be either narrow or unstable – meaning they either solve too few business queries to be truly exploratory, or they don't provide enough stability in the conformance and currency of data to be effective. In a competitive business, systems and mind-sets are in constant flux, and the evolution of dimensional models rarely keeps pace with this.

P 7: KS.docx - 7:13 [11. Describe the advantages or..] (46:47) (Super)
Codes: [Inmon type of philosophy - Families (3): Cyclical development, Methodology, Slowly changing dimensions]
No memos

11. Describe the advantages or disadvantages of maintaining dimensional models.

The dimensional modelling approach, at the logical level, can be used by multidimensional database or even flat files, not tied to any particular implementation. This model can drill down upto the finest granularity because of the fact table. The dimensional model may not fit some application because the way how data is looked, perceived by the user like sequencing of data and data definitions could be different. The implementation of dimensional model to the whole organization may not serve the purpose for some departments and raises important questions about the implementation strategy of data warehouse .

When examining the actual methodologies being practised, Table 6-14 reveals that 4 participants practised Kimball methodology, 4 practised a waterfall type of methodology and 4 participants practised agile methodologies, of which 2 did so in conjunction with the Kimball methodology and 1 was in conjunction with the waterfall methodology.

The data warehouse technology used by an organisation has an influence on the methodology used. The code "Technology reliance and dependency" occurs 5 times in 4 of the 11 responses. Code excerpt 16 below contains text from P3's and P8's responses which demonstrates that the limitations of a chosen technology influences the methodology.

Code Excerpt 16 – Technology reliance and dependency
P 3: DN.docx - 3:1 [Describe the methodology or de..] (4:5) (Super) Codes: [Agile type of methodology - Families (2): Cyclical development, Methodology] [Methodology linked to technology - Family: Methodology] [Technology reliance and dependency - Family: Methodology] No memos 1. Describe the methodology or development process that you go through when developing data warehouses and where maintenance fits into this methodology. A very limited process is used where the requirement is identified with the customer by the QlikView developer. Thereafter the developer would implement the request and get user signoff afterwards. A small request can take from 5 days and more. Maintenance and updates are done on request – small changes are done on the fly, whilst changes with a larger impact will be scoped and costed. **P 8: MN.docx - 8:16 [13. Describe the advantages or..] (138:141) (Super)** Codes: [challenge of DW methodology knowledge - Families (2): Challenges experienced, Dimensional modelling] [Technology reliance and dependency - Family: Methodology] No memos 13. Describe the advantages or disadvantages of maintaining dimensional models. Data clean up is quite a bigger issue with multidimensional models Not everyone knows how to create a proper dimension, so a lot of the advantage are thus removed Limited in SAP to 16 dimensions, which mostly is sufficient.

6.6.8.1 Conclusions drawn on Methodology

There has not been much exposure to methodologies other than the Kimball life cycle. Participants do not feel that any particular methodology is better than the other

as they have not been exposed to various methodologies. This could indicate that data warehouse development is still an immature sector in software development in South Africa. That being said, it is too early to draw a conclusion on this as 11 respondents are too few.

It was found that with some technologies, e.g. SAP, the technology drives the methodology as SAP deployment often includes a data warehouse deployment. This deployment is usually done as one project.

6.6.9 Theme 9: OLTP comparison

Questions 19, 20 and 21 aim to understand the participant's opinion of the difference between OLTP development and maintenance and DW development and maintenance. There were many participants who did not have any experience with OLTP systems and left these questions unanswered. Table 6-15 lists the codes created from the responses received.

	P2	P3	P4	P5	P6	P7	P8	P9	P 10	P 11	P 12	TOTALS
Availability higher priority with OLTP	0	0	0	0	0	2	0	0	0	0	0	2
DW used to ease the load on transactional system	0	0	0	0	0	0	0	0	0	2	0	2
More focus placed on perfective maintenance with OLTP	0	0	0	0	0	1	0	0	0	0	1	2
OLTP dev require more testing	0	0	0	0	0	0	0	0	0	1	0	1
OLTP forces users to change	0	0	0	0	0	0	0	0	1	0	0	1
OLTP given higher priority for maintenance	0	0	0	0	0	0	0	0	0	1	0	1
OLTP has simpler data retrieval	0	0	0	0	0	2	0	0	0	0	0	2
OLTP maintenance focus is on system architecture (hardware)	0	0	0	0	0	1	0	0	0	0	0	1
OLTP maintenance focuses on release management	0	0	0	0	0	1	0	0	0	0	0	1
OLTP maintenance less challenging as data analysis not required	0	0	0	0	0	1	0	0	0	1	0	2

	P2	P3	P4	P5	P6	P7	P8	P9	P 10	P 11	P 12	TOTALS
OLTP maintenance less challenging as reloads not required	0	0	0	0	0	1	0	0	0	1	0	2
Performance more important in OLTP	0	0	0	0	0	2	0	0	0	0	0	2
TOTALS:	0	0	0	0	0	11	0	0	1	6	1	19

Table 6-15 Count of codes per document for OLTP Comparison

6.6.9.1 Conclusions drawn on OLTP Comparison

When examining the codes created in this family, one notices that OLTP maintenance seems to be less challenging but takes preference as the transactional system is generally a mission critical one. The volumes of data are a major factor which differentiates OLTP maintenance from DW maintenance. High volumes of data leads to slow load times and requires better performance management.

.

6.6.10 Theme 10: Outsourcing

The code family of outsourcing contains many codes which are not in other code families. In Table 6-16 below, the only code that has already been discussed is “communication challenges of outsourcing” as this code is common to the communication code family and the challenges experienced code family.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Communication challenges of outsourcing	4	0	0	0	0	2	1	0	1	0	0	8
Different developers doing maintenance	1	0	0	0	0	0	1	0	0	0	0	2
High cost of outsourcing	0	1	1	0	0	1	0	0	0	0	1	4
Outsourced developers do more harm by focusing on getting the job done quickly	1	0	0	0	0	0	1	0	0	0	0	2
Outsourced developers have to spend time learning	1	0	1	1	0	0	3	1	1	0	0	8
Outsourcing negative	1	1	0	1	0	0	3	0	1	0	0	7
Outsourcing pose risk of losing knowledge of a system	0	1	0	0	0	0	0	0	0	1	0	2
Outsourcing positive	0	0	2	0	0	0	0	0	0	0	1	3
Totals:	8	3	4	2	0	3	9	1	3	1	2	36

Table 6-16 Count of codes per document for Outsourcing

The next code, “Different developers doing maintenance” is something that one might expect when discussing outsourcing. Not all participants alluded to this as many of the participants answered the questions in a written format, perhaps the question was not made clear enough to them. However, as shown in Code Excerpt 17 below, P2 did mention that outsourced consultants only do maintenance due to bad development:

Code Excerpt 17 – Different developers doing maintenance
P 2: AB.docx - 2:12 [Shakeel: You know, we’re more ..] (33:36) (Super) Codes: [Different developers doing maintenance - Families (3): Dimensional modelling, Methodology, Outsourcing] No memos Shakeel: You know, we’re more after your experiences. So remember that there’s no right or wrong answer – just talk about your experience. AB: Hardly ever maintenance done on a DW. Usually when something needs to be fixed due to bad development. When bad data that comes through then maintenance is done with reloading the data warehouse. It depends on When can you take down the source systems to do the reloads and how much impact there is on the source systems.

A more frequently occurring code is “Outsourced developers have to spend time learning”. Working on DW system for the first time does pose a challenge for many DW professionals. While P4 was generally positive about consultants, he did mention that it could take time for DW professionals to gain an understanding of the source systems as shown below in Code Excerpt 18.

Code Excerpt 18 – Outsourced developers have to spend time learning
P 4: MW.docx - 4:16 [23 From your perspective, what..] (165:167) (Super) Codes: [Outsourced developers have to spend time learning - Family: Outsourcing] [Outsourcing positive - Family: Outsourcing] No memos 23 From your perspective, what are the advantages and disadvantages of using consultants in DW/BI projects in terms of maintenance Generally, it’s easier to keep consultants focused on producing business value, with the drawback that it can take months on complex systems for individuals to arrive at a correct understanding of how the OLTP system produces and modifies it’s data.

When reflecting on DW professional's general outlook toward outsourcing, the negative comments outweighed the positive ones. P2 did mention that outsourced development could hinder maintenance at a later stage as shown in Code Excerpt 19.

Code Excerpt 19 – Outsourcing Negative
P 2: AB.docx - 2:21 [Shakeel: If you have experienc..] (110:112) (Super) Codes: [Outsourced developers do more harm by focusing on getting the job done quickly - Families (2): Communication, Outsourcing] [Outsourcing negative - Family: Outsourcing] No memos Shakeel: If you have experience as a consultant, talk about experience of DW dev/maintenance as being a consultant as opposed to being in-house. AB: Being in-house, you build it according to your standards and you're able to maintain it better. As a consultant, the focus is on getting the work done. It doesn't matter if the design was bad and the client could be left with a hard to maintain environment as you have difficult time constraints to manage.

6.6.10.1 Conclusions drawn on Outsourcing

The participants in this research are IT professionals and may be more critical toward outsourced co-workers than managers would be. The reason for this is that managers are more concerned with getting a project finished on time. P4, who is both a manager and an IT professional was more positive about outsourcing.

As there is a lack of DW resources, outsourcing is necessary. It cannot be avoided but should be managed with care.

6.6.11 Theme 11: Preventive/Perfective maintenance

Preventive and Perfective maintenance is a recurring theme in many code families. When there is a lack of preventive maintenance, it has many effects downstream and could cause the whole DW system to fail. We have thus already discussed many codes contained in this family. These include "challenge of duration for loading data", "Challenge of high volume addressed through perfective maintenance", "Unaware of Inmon's partitioning strategies".

The other prominent types of codes in the Preventive/Perfective maintenance code family is that of preventive maintenance not being performed. Table 6-17 contains the themes for this code family and there are 3 codes which relate to preventive maintenance not being performed. These are: “Lack of preventive/perfective maintenance”, “Low priority of perfective maintenance” and “Management does not prioritise perfective maintenance”.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Challenge of data integrity	0	0	0	0	1	0	0	1	0	0	0	2
Challenge of duration for loading data	0	0	1	0	0	0	2	1	0	0	0	4
Challenge of high volume addressed through perfective maintenance	0	0	0	0	0	1	1	0	0	0	0	2
Difficulty of using the technology	1	1	0	0	0	0	0	0	0	0	0	2
Hardware improvement to cope with volume	0	0	1	0	0	0	2	0	0	1	0	4
Lack of preventive/perfective maintenance	1	2	1	0	1	0	3	0	0	0	0	8
Low priority of perfective maintenance	1	0	0	0	0	1	1	0	0	2	0	5
Management does not prioritise perfective maintenance	0	0	0	0	0	0	0	0	0	1	0	1
Perfective maintenance is performed	0	0	1	2	2	2	0	2	1	0	1	11
Preventive maintenance monitoring occurs	0	0	1	0	0	0	0	0	1	0	1	3
Technology used to improve performance	1	0	0	1	0	0	0	0	0	0	0	2
Unaware of Inmon's partitioning strategies	2	1	1	1	1	1	1	1	0	1	0	10
Totals:	6	4	6	4	5	5	10	5	2	5	2	54

Table 6-17 Count of codes per document for Preventive/Perfective maintenance

Perfective maintenance is mostly done as a reactive process. Many participants did not seem to feel that this problematic. Code excerpt 20 contains responses for the code “Low priority on perfective maintenance”. The reasons for this given by P8 is that things are changing too frequently during development to focus on preventive or perfective maintenance. P11 felt that perfective maintenance should take a second priority to corrective maintenance. An opposing view is offered by P7 who felt that time, commitment and good process could be readjusted to address the lack of

preventive maintenance. The excerpt below contains responses from P7, P8 and P11. Code excerpt 20 below contains the text with these responses.

Code Excerpt 20 – Low priority on perfective maintenance
P7: KS.docx - 7:11 [9. Are there maintenance tasks..] (40:42) (Super) Codes: [Low priority of perfective maintenance - Family: Preventive/Perfective maintenance] No memos 9. Are there maintenance tasks that you feel should be included and if so, what factors are influencing the implementation of these Code refactoring on a periodic basis without affecting the functionality and need for testing. The factors that influence this are time, commitment and a good process. **P 8: MN.docx - 8:13 [10. Describe the regular maint..] (98:113) (Super)** Codes: [challenge of duration for loading data - Families (3): Challenges experienced, High Volumes, Preventive/Perfective maintenance] [High volume require better user communication - Families (2): Communication, High Volumes] [High volumes pose challenge - Families (2): Challenges experienced, High Volumes] [Lack of preventive/perfective maintenance - Families (3): Experience with Preventive/Perfective maintenance, High Volumes, Preventive/Perfective maintenance] [Low priority of perfective maintenance - Family: Preventive/Perfective maintenance] No memos 10. Describe the regular maintenance tasks that are performed on the data warehouse environment. So what's happening is on our side… well I don't know if you've heard of this "layered scale architecture" or it may be a SAP term. There's this architecture that makes the system more effective – e.g. makes sure you don't duplicate data and it's a very specific way of design and development. So we've got the system, but we're actually going back and redesigning the DW and that is like a maintenance task that we get a project team and they analyse and tell us; "listen, this cubes running ineffectively. Your dimensions or keys are wrongly defined" and we will get 10 days to go fix it. Shakeel: So, it's a very reactive process? Or is something that's done as part of development proactively. MN: so that's why we have this technical authority now. We go through this to ensure our design is ready from the start. But you know what, since I've been there, they've changed the design and architecture 3 times and I have conveniently ignored them because I didn't have time to do it. So lucky for me [that I didn't have to change things 3 times]. You see, they also learn this things – it's something new that SAP comes out with, like a new acronym. And then people feel like they have to implement it without realising that it's inefficient and stupid until someone like me comes in and tells them that I refuse to design something that way because it makes no sense. So now, after all these go-lives, there's a 9 month remediation project that's going to happen where they're going to fix the system effectively because the design in certain places is just not great.

Shakeel: Is the remediation due to users complaining?

MN: No, it's because our database runs out of space easily or it runs slow or when we have a problem and we have to reload, we lose 3 days instead of if it was designed effectively, we would lose 2 hours. So, yes, the business would complain about them losing 3 days but we do give them fair warning and explain why it happened- which is usually due to bad data entry on the user's side.

So, reloading generally means reloading data for 6 months which is generally like 200 million records. It shouldn't take 3days – it should only take half a day. So that's why we're improving the system – to make support and rollout easier. Cos we have to rollout in 17 different countries – so the improvements will makes that process easier, and make the system more understandable, and so forth

Shakeel: So that's quite challenging – rolling out?

MN: Oh Ja, cos you need to re-test everything. So now, business doesn't understand why they have to re-test something that has been working. It's painful.

P11: GW.docx - 11:11 [10. Describe the regular maint..] (36:41) (Super)
Codes: [Low priority of perfective maintenance - Family: Preventive/Perfective maintenance]
No memos

10. Describe the regular maintenance tasks that are performed on the data warehouse environment.

The main ones are on the ETL system – little bugs. Something like someone switches off a system which the ETL system depends on.

Shakeel: What about things like index building and performance oriented tasks?

GW: That's never really too difficult. If your design is right up front, then performance isn't too difficult to troubleshoot. Ultimately, it's about finding out why a report or process is taking a long time and then fixing it.

Shakeel: So it's a very reactive task?

GW: Ja, performance is second. In order of maintenance, it probably goes like: fixing up the ETLs if the break for whatever reason or fixing the performance on the ETLs if they get broken and then lastly looking at user performance issues.

There are also many occurrences of codes which indicate that preventive maintenance is part of the data warehouse operation. These include "Perfective maintenance is performed", "Preventive maintenance monitoring occurs". The combined frequency of these codes is 14 out of a total of 54 total code occurrences for this family. This demonstrates that preventive maintenance is being considered in many teams or organisations.

One somewhat common approach to dealing with preventive and perfective maintenance issues is to upgrade the hardware. In some instances, this may make sense as hardware has become more affordable than the cost of resources. In general, this is not considered a good practice. In Code Excerpt 21 below, we see that P4 admits that it is not an ideal way to deal with the problem.

Code Excerpt 21 – Hardware improvement to cope with volume
P 4: MW.docx - 4:12 [16 How do you overcome the cha..] (129:132) (Super) Codes: [Hardware improvement to cope with volume - Families (2): High Volumes, Preventive/Perfective maintenance] [High volumes pose challenge - Families (2): Challenges experienced, High Volumes] No memos 16 How do you overcome the challenge of maintaining high volumes of data At <1TB per database instance, we have haphazardly keeping monthly on-line archives on our SAN. In terms of moving volumes of data… as per answer 15, we don't really deal with it [well] – it's a problem for us.

6.6.11.1 Conclusions drawn on Preventive/Perfective maintenance

Some DW practitioners do not see value in spending time on preventive maintenance until it becomes a problem. This could be a symptom of lack of knowledge around the methodology of data warehousing as observed in Section 6.6.8.

The interpretive questionnaire did not delve into the reasons behind where preventive maintenance was or was not performed or the maturity of the systems that it was performed on. In order to understand the motivation behind performing perfective maintenance, one might be tempted to compare the experience of the participants who answered affirmatively to preventive maintenance being performed and the experience of the participants who did not feel that preventive maintenance was necessary. This comparison, however could be misleading as there might be other factors which influence the desire to perform perfective maintenance such as

maturity of the system, maturity of the DW warehouse team, project schedules or resource availability.

The notion of improving performance by upgrading hardware is an attractive one. However, this solution is generally temporary and new hardware is generally budgeted after a few years. If growth of a system becomes exponential, bad design or lack of performance considerations in the data warehouse will be exposed, even on the best hardware.

6.6.12 Theme 12: Training

Table 6-18 below lists the codes for the family training. Many codes were included in the code family of training as there are many facets of a data warehouse system that involve training. For example, the code “Lack of communication between developer and user” is also included in the family training as informal training occurs between developers and users when they communicate.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Challenge of changing user mind-set to use DW	0	0	0	0	0	0	0	0	0	2	0	2
Challenge of learning new system	0	0	0	0	0	0	1	0	0	0	0	1
Difficulty of using the technology	1	1	0	0	0	0	0	0	0	0	0	2
Direct contact between business and team	0	0	2	0	2	0	0	0	0	0	0	4
Effectiveness of Developer training	1	0	0	0	0	0	0	0	0	0	0	1
Lack of communication between Developer and User	2	1	0	2	0	2	3	0	1	1	1	13
Learning occurs under pressure	0	0	0	0	0	0	1	0	0	0	0	1
Maintenance requires experienced professionals	0	0	0	0	0	0	1	0	1	0	0	2
Priority on understanding business rules	0	0	0	0	0	0	0	2	0	0	0	2
Support activity is granting of access to users	0	1	0	0	0	0	0	0	0	0	0	1
Team knowledge leveraged	0	0	0	1	1	0	0	0	0	0	0	2
User Interface challenges	0	0	0	0	0	0	0	0	2	0	0	2
User training occurs	0	0	0	0	0	0	0	0	1	0	1	2
Users need re-training as DW system not used regularly	0	0	0	0	0	0	0	0	1	2	0	3
Totals:	4	3	2	3	3	2	6	2	6	5	2	38

Table 6-18 Count of codes per document for Training

Code excerpt 22 below demonstrates that communication between end users can assist with end user training. P4 explains that end users contact developers to ask questions about reports. One can imagine that during this communication, knowledge is transferred which is a form of training.

Code Excerpt 22 – Hardware improvement to cope with volume
P 4: MW.docx - 4:22 [9 How are the end users suppor..] (87:90) (Super) Codes: [Direct contact between business and team - Families (2): Communication, Training] No memos 9 How are the end users supported? We have dedicated IT teams who provide most of the end user support. When end users have queries about the reports produced for them, they communicate with the BI team in the same manner that stake holders do: At a time and through a medium that suits them.

When examining end user training, it is interesting to note that that users might need to repeat training on a DW system as a DW system is not used on a regular basis, as opposed to transactional systems which is used daily. The lack of usage of the DW system is related to the code "Challenge of changing user mind-set to use DW". Users are often resistant to change and require encouragement to switch from operational reports to DW reports and analysis.

Code excerpt 23 contains text from P2 and P8 which pertains to training of developers. P2 felt challenged by using development software even after attending training. P8 discusses the challenges of learning a new system under pressure and feels that the only way to learn a system is to get your hands dirty.

Code Excerpt 23 – DW professional training
P 2: AB.docx - 2:1 [Shakeel: Talk about some of th..] (2:3) (Super) Codes: [Difficulty of using the technology - Families (3): Documentation, Preventive/Perfective maintenance, Training] [Effectiveness of Developer training - Family: Training] [Technology reliance and dependency - Family: Methodology] No memos Shakeel: Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced and how you deal with handling of these challenges. AB: Challenges experienced where mostly understanding of a tool even with relevant training and applying the tool to different environments. **P 8: MN.docx - 8:5 [5. How do you address the chal..] (61:64) (Super)**

Codes: [Challenge of learning new system - Families (2): Challenges experienced, Training] [Documentation usefulness negative - Family: Documentation] [Learning occurs under pressure - Family: Training]
No memos

5. How do you address the challenge of working on a data warehouse that you have not worked on before?

I rarely work on different warehouses. And with the Microsoft one, I'm a little out of my depth when it's get's started.

Shakeel: I think the question is more on, when you joined company X, how did you learn the new warehouse.

MN: Oh… what happens normally when you're in SAP is that you don't have a window period of getting comfortable. You get thrown into the deep end. IN the end, the reports are easy, you can easily trace it back. So, I got there when it was go live. That's most probably the worst situation to start a new project because they've been developing for 2 and a half years and I pitch up and I'm the only one that's responsible for financial reports and I have no idea what they're on about. So then the give me documentation and I pretend I'm going to read it and I never do. I only learn a new system when there's problems on a report and I go look at what the issue is. I will then work the report back to the data flow to the operation system tables and work with the functional guys to figure out what's going on. I don't do documentation. It's not how I work at all. I just get my hands dirty.

6.6.12.1 Conclusions drawn on Training

It seems to be more often the case that developers do not interact with users. This hinders user training. Lack of knowledge on the use of a system and lack of a professional support structure would discourage adoption of a new DW system. These are reasons for developers to embrace user support, especially after initial deployment.

The frustrations experienced by developers with a DW system could be related to the high pressure experienced in the data warehousing portion of a project. This pressure is amplified when there is a lack of skilled resources and one does not have experience with a tool as in the case of P2 in Code Excerpt 23. As described by P8 in Code Excerpt 23, there is a sudden push for DW development after everything else has completed and one is required to do both support on reports deployed and new development. These kinds of frustrations relate to methodology

6.6.13 Theme 13: Use of tools

Questions 24 and 25 where directed toward understanding how prevalent useful development tools were in data warehouse maintenance. As can be seen from Table 6-19 below, most participants indicated that maintenance tools are not being used and documentation tools are not being used.

	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	TOTALS
Development tools not well suited for maintenance	0	0	0	0	0	0	0	0	0	1	0	1
Documentation tool used	0	0	1	0	0	0	0	0	0	0	0	1
Documentation tools not being used	0	1	0	3	1	1	2	2	1	0	1	12
Maintenance tools not being used	0	1	1	1	0	1	1	1	1	1	1	9
Technology used to improve performance	1	0	0	1	0	0	0	0	0	0	0	2
Tools used for error handling	0	0	0	0	0	1	0	0	0	0	0	1
Visual BI aids the relationship between business and DW team	0	1	0	0	0	0	0	0	0	0	1	2
TOTALS:	1	3	2	5	1	3	3	3	2	2	3	28

Table 6-19 Count of codes per document for Use of tools

6.6.13.1 Conclusions drawn on Use of Tools

From the literature review conducted, it seems that there were more tools available for OLTP development than there were for data warehouse development. Maintenance tools were also not commonplace. As such, it would be surprising to find a prevalent use of tools for data warehouse development.

Documentation tools, on the other hand are more available however these tools are not being used. The reason for this could be, as discovered in Section 6.6.5 that documentation tools produce verbose documentation which is not useful.

6.7 Conclusion

In this chapter, the research mode, research methods and research process were discussed and the research conducted was analysed.

Due to the cyclical nature of the research, where detailed understanding leads to a higher level of understanding and vice versa, the research mode chosen was hermeneutics.

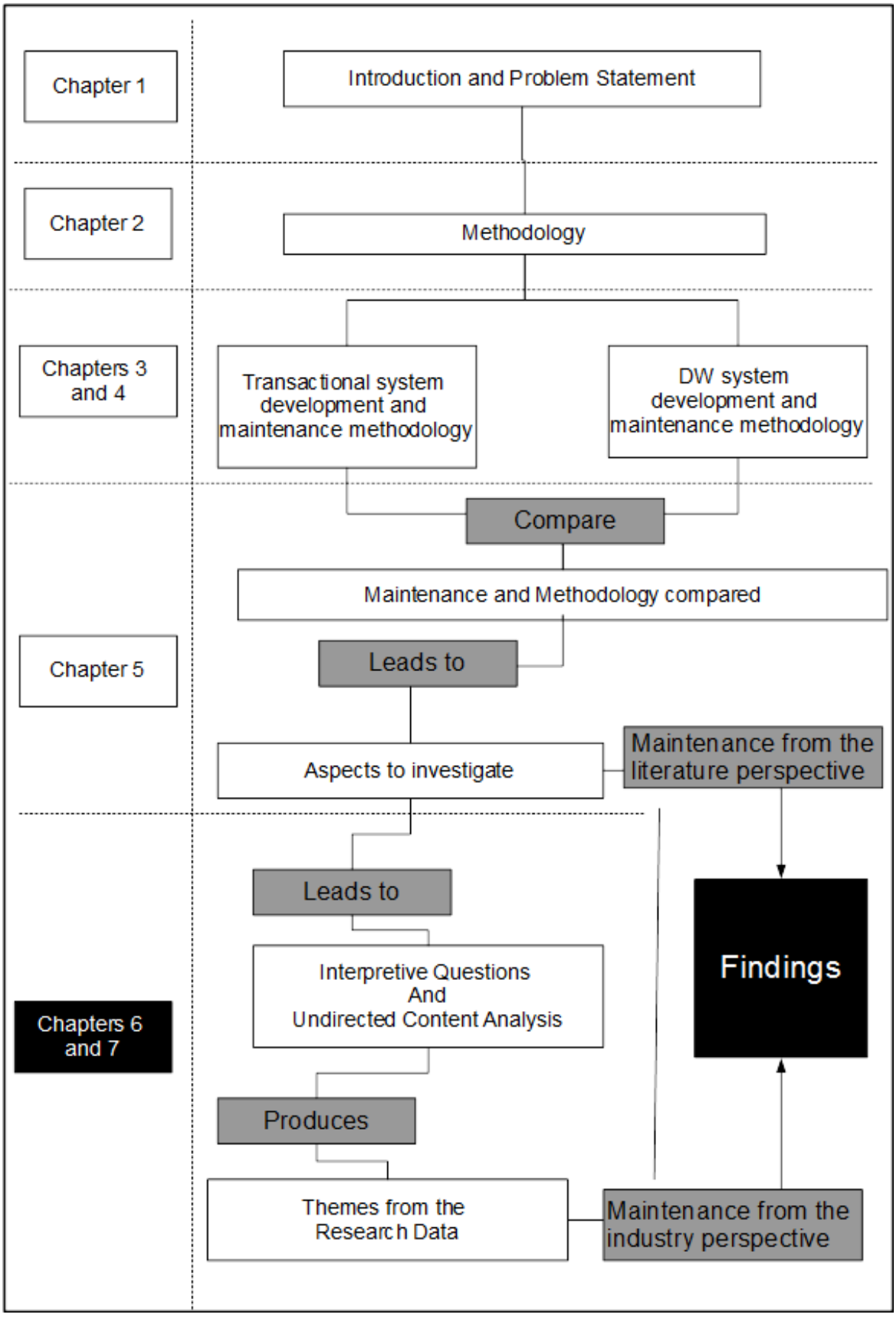
Before beginning analysis, the research process consisted of developing an interpretive questionnaire, recruiting participants to answer the questionnaire, transcribing the interviews conducted and then coding the transcribed interviews.

In Chapter 2, it was decided that content analysis would be the research method. Zhang and Wildemuth (2009: 3) define 8 steps to performing content analysis and these were the steps followed. As the research conducted was based on literature surveys conducted, analysis began with directed content analysis. After testing the effectiveness and correctness of this approach, it was found that undirected content analysis allowed a more complete analysis.

As presented in the diagram preceding this chapter, Chapter 6 contained an analysis of the interpretive questionnaire responses by using undirected content analysis.This produced themes in the form of code families from the research data. To this end, the undirected content analysis employed text coding.

The code families were analysed and conclusions were drawn on each of them. That concludes the practical research portion of this dissertation.

The next chapter is the final chapter. It explores the conclusions drawn in this chapter in Section 6.6, concludes the research conducted and gives recommendations.


Chapter 1
Introduction and Problem Statement
Chapter 2
Methodology
Chapters 3 and 4
Transactional system development and maintenance methodology
DW system development and maintenance methodology
Compare
Maintenance and Methodology compared
Chapter 5
Leads to
Aspects to investigate
Maintenance from the literature perspective
Leads to
Interpretive Questions And Undirected Content Analysis
Findings
Chapters 6 and 7
Produces
Themes from the Research Data
Maintenance from the industry perspective

Chapter 7
Conclusions and Recommendations

7.1 Introduction

This is the final chapter of this dissertation. This research aims to compare the challenges of maintenance of data warehouses and business intelligence systems experienced in a South African context with the challenges of maintenance and support of transactional software systems.

In Chapter 6, the previous chapter, the research conducted in the form of an interpretive research questionnaire was discussed. As it was discovered that most participants did not have experience in a transactional software system, the comparison was made against the literature surveyed in Chapter 5 and previous chapters.

In this chapter, the research performed is evaluated in Section 7.2 and in Section 7.3, the themes which were coded in the research process is compared to the research aspects that serve to summarise the literature surveyed.

7.2 Research aims and objectives revisited

The research aims and objectives were defined in Section 1.4. These are revisited here with a discussion on how these aims and objectives where achieved.

The main objective of this dissertation is to compare the challenges between maintenance of transactional systems and the maintenance of data warehousing systems. Achieving the sub-objectives below ensures that this overarching aim and objective is achieved. The discussion that follows lists these sub-objectives and how they were accomplished.

1. To understand transactional software development and maintenance methodologies as well as to understand challenges of maintenance of transactional systems by doing a literature review.

Transactional software development and maintenance methodologies were reviewed in Chapter 3. The ISO standard for software maintenance was summarised as a practical approach to maintenance in Section 3.3.1. The challenges in software maintenance, as encountered in the literature review were discussed in Section 3.3.2 and the approaches, as encountered in the literature; to addressing these maintenance challenges were reviewed in Section 3.3.3. The challenges of transactional software maintenance were again reviewed and summarised in Chapter 5 where these challenges were compared to those of data warehouse maintenance challenges.

2. To understand data warehouse development and maintenance methodologies as well as to understand challenges of maintenance of data warehouse systems by doing a literature review.

Data warehouse development and maintenance methodologies were reviewed in Chapter 4. These were divided into the approach recommended by Bill Inmon in Section 4.3.1 and the approach recommended by Ralph Kimball in Section 4.3.2. The challenges in data warehouse maintenance, as encountered in the literature review were discussed in Section 4.3.1.4 and 4.3.2.9. The challenges of data warehouse maintenance were again reviewed and summarised in Chapter 5 where these challenges were compared to those of transactional software development maintenance challenges.

3. To compare challenges of transactional software maintenance and the challenges of data warehouse maintenance by examining the literature reviews conducted.

The challenges of transactional software maintenance and challenges of data warehouse maintenance were compared in Chapter 5. The comparison was divided into four major empirical research areas: maintenance methodology, perfective and preventive maintenance, corrective maintenance, adaptive maintenance.

4. To understand the data warehouse maintenance methodologies and maintenance challenges being practiced in South Africa and to understand whether these practices and the management of these challenges correlate to data warehousing methodologies.

This has been done by conducting an interpretive study on data warehouse maintenance in South Africa. Here an investigation has been undertaken into maintenance methods being practiced by data warehouse maintenance professionals. The literature reviews and comparison performed in Chapters 3, 4, and 5 were used as a basis to develop an interpretive research questionnaire and professionals from the data warehouse and software development industry participated in answering this questionnaire. The questionnaire was used a guide during interviews and questions were asked in a semi-structured fashion. The responses were then coded using undirected content analysis. The findings of this study were presented in Section 6.6.

5. To present the findings on the challenges of maintenance and support of a data warehousing environment compared the challenges of maintenance and support of transactional systems.

In Section 7.6, information sourced from literature and the information sourced by the analysis of the responses to the interpretive research questionnaire are related to form conclusions of the study.

7.3 Research evaluation

As discussed in Chapter 2, this research has not been hypothesis driven and hence it did not follow a positivistic philosophical perspective. It has been driven by social interaction in the form of interviews and written questionnaires and has followed an interpretive perspective. The mode of analysis, as discussed in Chapter 6 is hermeneutics which deals with the interpretation of text by cycling between understanding smaller portions of text to understanding the larger portion.

When evaluating interpretive research in information systems, Klein and Myers (1999: 67) have developed a set of interdependent principles to follow. These were discussed in length in Chapter 2. These principles are: the principle of hermeneutic cycle, the principle of contextualization, the principle of the interaction between the researcher and the subjects, the principle of abstraction and generalization, the principle of dialogical reasoning, the principle of multiple interpretations and the principle of suspicion.

7.3.1 The principle of hermeneutic circle

During this research, understanding was developed by first targeting individual responses to questions, then examining the text in its entirety. As text was re-examined as an entire unit once again, similar ideas were grouped together to form code families.

During the analysis, these codes are analysed as individual codes, as part of a code family, as part of a single interview or as part of the entire text.

7.3.2 The principle of contextualization

The context of this research is limited to experience of data warehouse software professionals within South Africa. The backgrounds and experience of participants were noted and these were referred to during the comparison process.

For example, in section 6.7 it was noted that participants who worked in a SAP technology environment felt that chosen technology was the driving factor in the choice of methodology. It was also noted in the given context, our limited number of participants had not been exposed to a variety of data warehouse methodologies, indicating that participants in our given context have limited experience.

The context of our participants is the majority of them do not have transactional system development experience. This led to a re-orientation of the focus of the empirical study from comparing OLTP and DW maintenance to investigating the DW maintenance practices of professionals from a DW and OLTP methodological standpoint.

7.3.3 The principle of interaction between the researchers and the subjects

In order to prevent the researcher from influencing responses by participants, an introduction to the interpretive questionnaire process was provided. This introduction, along with the interpretive questionnaire is included in appendix A. Part of the purpose of the introduction was to set participants at ease, letting them know that the interview is anonymous and that there are no correct or incorrect answers to the questions. The introduction was discussed with telephonic participants at the beginning of the interview and was included as the first page of the interpretive research questionnaire to written responses.

7.3.4 The principle of abstraction and generalization

The entire text body of a participant's response was coded. Codes were grouped together into code families. Each code family was discussed in the context of data warehouse practitioners within a South African context as well as compared to the literature review which had been performed and reported on in the preceding chapters. Many responses of participants correlated even though they came from different organisations. Hence, some generalisation in the South African context is possible.

7.3.5 The principle of dialogical reasoning

In order to follow this principle, Klein & Myers (1999: 76) emphasize that the researcher needs to make philosophical assumptions transparent to the audience as well as to himself or herself, as well as the interpretive methods chosen and their strengths and weaknesses.

The researcher was motivated to take on the study partly due to frustration with personal experience of DW development and maintenance practises in the industry. This personal experience led to the view that industry professionals, do not implement DW methodologies as they were designed which led to OLTP type of maintenance problems in the DW environment.

This study allowed the researcher to focus on the fundamental differences between OLTP and DW development and maintenance methodologies. The researcher is now able to present a grounded conclusion on the matter based on literature review and the interpretive study. This conclusion is developed in Section 7.4 and summarised in Section 7.5.

7.3.6 The principle of multiple interpretations

Amongst the participants were those who focussed on analysis and support, those who had management experience and many with experience in differing experience. As such there were many responses with differing views and sometimes opposing views.

Examples of opposing views include views on documentation and preventive maintenance. It was surprising that many participants did not find documentation useful and it was reasoned that outdated documentation could be the cause of this. Whilst many participants indicated that there was significant preventive maintenance being performed, many also indicated that this was an area that required attention. It

was noted that there might be many reasons for this such as project schedules, experience, budget, etc.

7.3.7 The principle of suspicion

The majority of participants have a data warehouse development background. There are a few of these that have management experience and experience as business analysts or customer support professionals. The bias contained in the responses often correlate to the type of experience of the participant. Bias was also influenced by the technology with which the participant was more involved in. During the analysis process, these biases were noted where it was interpreted to have an influence on the response provided.

7.4 Research discussion

In this section, the themes developed through coding is re-examined. Each theme is linked back to the table of research aspects developed in Chapter 5 and summarised in Table 6-1. This is done by describing how the theme which emanated from the coding process relates to the aspects which emanated from the literature review process.

7.4.1 Theme 1: Challenges experienced

From the analysis of the interpretive research questionnaire on this theme, presented in Section 6.6.1, the challenges of high volumes and communication are the most ubiquitous. High volumes is a challenge closely related to preventive maintenance as the challenge of high volumes is managed through preventive and perfective maintenance. Communication is a theme on it's on own and is discussed in the next section.

The most common challenges experienced were related to preventive maintenance and communication. From the literature reviewed in Chapters 4 and 5, as summarised in Section 6.2, it is noted that the priority of preventive maintenance is

an aspect of data warehouse maintenance that warrants more time and attention. Further to this, the literature reviewed in Chapters 4 strongly recommends that data warehouse development is continual and on-going and that maintenance should not be treated as a phase in a development process.

Table 7-1 below shows the aspects that relate to the code family of challenges experienced and a short description of what the relationship is. This is a very broad code family and it relates to most empirical research aspects. The codes that belong to this family and their counts are listed in Table 6-9.

1	*Cyclical or iterative methodologies allow requirements to be added in new iterations.* In sequential development methodologies, requests that are made after deployment are considered to be maintenance requests. In a DW system, which follows cyclical development, these requests should be considered to be requests which can be addressed in a new iteration.
2	*Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.* This aspect deals with communication challenges, learning a new system challenges and documentation challenges.
3	*Use of a steering committee (which includes representation from the source systems, business management and IT).* A steering committee assists in gaining understanding between all departments involved in the data warehouse process. A steering committee should have a key role to play in budgeting and time constraints of a project hence relates to the challenge of resource and time constraints.
4	*Priority of preventive maintenance and when it is performed.* This aspect is related to the challenge of high volumes, challenge of duration for loading data and the challenge of data integrity.
5	*Use of metadata including documentation.* Updating a data is a challenge hence this aspect relates directly.
6	*Use of dimensional models.* Dimensional models make learning a new system easier, hence this aspect relates to the challenge of learning a new system. Dimensional models are specific to DW methodology hence this aspect also relates to this challenge.
7	*Addressing challenges of high volumes of data.* This relates directly to the challenge of high volumes and indirectly to the challenge of duration for loading data.
8	*Partitioning scheme usage and strategy.* Partitioning also relates to the challenge of high volumes and the challenge of duration for loading data.
9	*Training of users and cross skilling of developers.* The use of outsourcing hinders the cross-skilling of developers hence this aspect indirectly relates to the use of outsourcing. The user interface of a system affects the willingness of users to use a system hence this aspect also relates to the User Interface challenge. The most direct relation that this aspect has is with challenge
10	*Is maintenance being done primarily by junior developers?* Junior developers require more time to learn a new system and require a higher level of communication. This research aspect relates to the challenges of learning a new system and communication.

11	*Use of outsourcing.* Outsourcing relates to communication challenges and this was coded as an addition to the communication challenges.
12	*Involvement of development team on maintenance tasks.* This aspect relates to the challenges of resource and time, methodology, team communication and outsourcing communication.
13	*Logical hardware separation.* This aspect relates to preventative maintenance hence the challenges of high volumes and data load duration pertain.

Table 7-1 Research aspects related to challenges experienced

From the information sourced from the literature and the analysis of data, the researcher feels that more time should be spent on continual maintenance. In OLTP systems, corrective maintenance is a phase which ends, however preventive maintenance and perfective maintenance occurs regularly as OLTP systems tend to be mission-critical. As, this is not the case with a data warehouse, data warehouses do not enjoy the same level of priority.

7.4.2 Theme 2: Communication

From the analysis of the data gathered through the interpretive research questionnaire in Section 6.6.2, it is distinct that a data warehouse steering committee is often not used. Another common observation is that there is difficulty in communication between developers and users.

Table 7-2 contains an excerpt from the interview with P8 who is the lead developer for the data warehouse. This excerpt demonstrates a separation of the user from the developer. Further, because a different team is involved in maintenance, the lead developer is unaware of requests from users for changes or enhancements unless she is asked for assistance. The methodology described by P8 also demonstrates that cyclical development is not being practised.

MN: Ja, I hear what you're saying. See, you're in the educational side of it. In all honesty, I just do what I have to get the people off my back so. If I can explain to you maybe how the project works. I'm working for (Company X) and we're implementing a global system but every time we go a new country we enhance the global system with what they need. There's an entire red tape process of analysis. They'll have an entire team that just does analysis and that's something done by an outside team and I would be used for advice. Then you estimate and so forth but maintenance….

Shakeel: Um… would you describe as sequential or is analysis and maintenance done back and

forth. **MN: No, we develop the entire system and then maintenance is done by another team. We get the specifications to the T and then we develop so there's no going back and forth**

Table 7-2 Participant response indicating separation from users

Different methodologies handle communication differently as noted in Chapters 4 and 5. It should be noted that the shorter release cycles associated with cyclical development allows for more frequent communication with the user community and allows for the user community to add requirements as these emanate from the use of the data warehouse system. The Kimball life cycle, discussed in Chapter 4 strongly recommends the use of a steering community which is not being used as indicated by many of the participants. The theme of Cyclical Development is discussed in the next section.

This is evident from a literature perspective as the summary of the literature review presented in Table 6-1 shows. It is clear that communication challenges could emanate from the 8 of the 14 research aspects in this table.

Table 7-3 below lists these along with a description of how these research aspects relate to communication challenges. The aspect number refers to the original aspect number as listed in Table 6-1.

1	*Cyclical or iterative methodologies allow requirements to be added in new iterations.* With methodologies which have long iterations, requirements need to be communicated and finalised in the early stages.
2	*Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.* If the original team stays on, there does not need to be a hand-over which requires documentation and communication between development teams
3	*Use of a steering committee (which includes representation from the source systems, business management and IT).* A steering committee assists in gaining understanding between all departments involved in the data warehouse process.
5	*Use of metadata including documentation.* Documentation prevents repeated questions around the same topic which aids communication.
6	*Use of dimensional models.* One of the advantages of Dimensional models is that they are easy to understand and easy for business people to relate to. This understanding assists communication between the business and development teams.
9	*Training of users and cross skilling of developers.* Training and cross skilling allow for better

	understanding amongst development teams and between development teams and users.
11	*Use of outsourcing.* When maintenance teams are outsourced, care needs to be taken around communication. As members of the outsourcing team and the internal team have not built up a rapport with each other, communication can be more challenging.
12	*Involvement of development team on maintenance tasks.* If the development team is heavily involved in maintenance as well, there is less of a need for communication between the development and maintenance teams.

Table 7-3 Research aspects related to coding theme "communication"

Similarly, the use of outsourcing could be linked to methodology. The data warehouse methodology literature surveyed in Chapter 4 cautions against the use of outsourcing yet the information gathered from the interpretive research questionnaire shows that outsourcing is used extensively.

The researcher concludes that this deviation from recommended data warehouse methodology is due to a lack of skilled data warehouse resources available or a lack of understanding of data warehouse methodologies and principles by the software industry in South Africa.

7.4.3 Theme 3: Cyclical development

From the interpretive study is was known that half of the participants indicated they do not practice cyclical development, Only 1 participant indicated positively that cyclical development is being practised. This indicates that sequential development such as the waterfall methodology discussed in Section 3.2.1 is being practised. This methodology is one commonly used during transactional system development.

In terms of literature, cyclical development relates to the first aspect of Table 6-1. Cyclical development allows for reprioritization of requirements. Inmon *et al.* (2008: 9) reasons that users only begin to appreciate and understand the value of the data warehouse after they begin using it.

As cyclical development is not a common practice amongst the participants, it is concluded by the researcher that the absence of cyclical development in a data warehouse development and maintenance leads to other related challenges such as

communication as discussed in Section 7.4.2 and preventive maintenance challenges as discussed in Section 7.4.1 and Section 7.4.11. This is mainly because more meaningful requirements with a higher priority for the data warehouse may only surface after initial deployment.

7.4.4 Theme 4: Dimensional models

From the research conducted and summarised in Section 6.6.4, we know many participants do use dimensional models however, from the same section, we also know that Kimball's bus matrix is not being utilised. This implies that the Kimball methodology is not being precisely followed.

The research conducted and summarised in Section 6.6.1 also led us to the finding that high volumes is a widespread challenge. This challenge is related to the use of dimensional models as the of dimensional models help reduce volume of data. This is especially true when a bus matrix of processes and dimensions is followed and conformed dimensions are identified, ensuring that data is not unnecessarily duplicated.

It is also conceived from the interpretive research conducted that steering committees are not being formed for the development of data warehouses. This is another strong recommendation by Kimball which is not being heeded. The use of a steering committee plays a key role in preventing duplicate data being created in multiple departmental data silos.

In Section 6.6.4, it was reasoned that even among the software practitioners who follow a Kimball data warehouse life cycle, not many are aware of the uses of a Kimball bus matrix and the use of conformed dimensions. This lack of application of conformed dimensions may be associated with many research aspects from Table 6-1. Table 7–4 below lists these along with a description of how these research aspects relate.

1	*Cyclical or iterative methodologies allow requirements to be added in new iterations.* The dimensional bus matrix and cyclical iterations work well together. Each iteration of a data warehouse project allows the bus matrix to be re-examined thereby decided on the next business process to include in the data warehouse and the next set of dimensions required.
2	*Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.* Following a dimensional modelling approach should imply that the data warehouse is constantly growing hence there is no definite handover of maintenance.
3	*Use of a steering committee (which includes representation from the source systems, business management and IT)*. The steering committee decides on priorities of what should be added to the data warehouse next. These priorities should be based on a long term plan which emanates from the Kimball bus matrix.
6	*Use of dimensional models.* This research aspect was included as the proper use of dimensional models and planning around dimensional models plays a key role in the data warehouse life cycle.
7	*Addressing challenges of high volumes of data.* As dimensional model planning assists in reducing duplicate data, it plays a role in addressing the challenges of high volumes of data.

Table 7-4 Research aspects related to coding theme "dimensional models"

When performing this data warehouse development, the surveyed literature from Chapter 4 strongly recommends that original development team should always be available to continue where the previous cycle ended. In practice, data warehouses are built to fit a project time constraint and maintenance is handed over to a different team. This practice draws a parallel to transactional systems practices using the waterfall method where maintenance is a phase that occurs in a fixed time period. The waterfall methodology was reviewed in Section 3.2.1.

A data warehouse steering committee is a requirement to the data warehouse life cycle. They assist in assessing the current data warehouse from both a technical point of view and a business point of view. It shall be up to the steering committee to decide on the next business process to implement in the data warehouse. A steering committee in a large enterprise also prevents the creation of departmental data silos which leads to inconsistent reporting and duplication of data. Duplication of data exacerbates the challenge of high volumes of data. When incorporating the use of conformed dimensions in a data warehouse, the risk of duplicate data is reduced or planned for. As steering committees are not being formed in practice, one may perceive the development methods followed to be similar to transactional system development where requirements are gathered to fulfil a specific solution.

It is concluded by the researcher that it is important that a cyclical methodology be followed when implementing a dimensional data warehouse so that new dimensions and business processes can be added to the data warehouse according to adjusted requirements and time constraints. In comparison, the requirements of transactional systems are known at the onset of a project allowing for time and resource and architectural constraints to be calculated prior to usage of the system.

7.4.5 Theme 5: Documentation

The research conducted on the theme of documentation was discussed in Section 6.6.5. A challenge of documentation is keeping it updated hence many participants found that documentation was not useful or unavailable. It was also observed in Section 6.6.7 and that metadata is not fully understood hence is not being used Metadata aids documentation, especially when tools are used to read the metadata and generate documentation however, in Section 6.6.13 it was detected that software tools are also not being used in data warehouse maintenance.

Lack of documentation or bad documentation creates communication challenges therefore the theme of documentation is strongly related to the theme of communication. Many participants indicated that documentation is not useful and many indicated that it is non-existent. These are the main challenges related to documentation. Table 7-5 shows the research aspects that relate to the coding theme of documentation and these challenges.

2	*Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.* If the original development team hands over to a different team for maintenance, then the team receiving the handover would require documentation and find it useful.
5	*Use of metadata including documentation.* Metadata allows for automated creation and updating of system summaries hence assist in the creation of documentation. However, this type of summaries need to be supplemented by some sort of user-created documentation as
6	*Use of dimensional models.* Dimensional models are easier to understand and have a recognisable structure. This has a ripple effect to the documentation as the documentation is easier to understand and follow.

9	*Training of users and cross skilling of developers.* Documentation assists in training. If developers are to be cross skilled and users are trained, then documentation will be more useful
11	*Use of outsourcing.* If an enterprise makes extensive use of outsourcing, then there is usually a requirement that documentation be provided for the product being deployed. This might seem like an advantage to outsourcing however if the documentation is outdated or incomplete, there is often no-one left to assist rendering the documentation un-useful.
12	*Involvement of development team on maintenance tasks*. If the maintenance team is involved in maintenance, then there would be verbal assistance to supplement documentation.
14	*Usage and availability of maintenance tools*. Some maintenance tools can generate documentation based on the system metadata.

Table 7-5 Research aspects related to coding theme "documentation"

In Section 4.3.2.9, it was discussed that a data warehouse project should not be handed over to a maintenance team. Keeping on the data warehouse developers to perform maintenance would reduce the need for development documentation as the original developers should be able to assist newer team members.

It is concluded by the researcher that the use of metadata and maintenance tools will allow automated updating of documentation. Documentation has positive impact on communication, cross skilling of developers and training of users. Cross skilling allows for multiple team members to perform various different types of tasks. However, verbose documentation is also unappreciated and unused by developers. Therefore, there needs to be a balance between documentation and face to face communication between team members. As recommended by the literature, original development team members should remain on the project for future iterations hence supplementing the need for comprehensive documentation.

7.4.6 Theme 6: High volumes

As discussed in Section 7.4.1, high volumes of data is one of the most widespread challenges among the research participants. Analysis during this code family discussed in Section 6.6.6 reveals that participants are unaware of logical partitioning strategies and that many participants do perform some level of preventive or perfective maintenance.

Intuitively, concerns surrounding volumes of data lends oneself to look at preventive maintenance literature. This relates to the research aspects 4, 7, 8 and 13 of Table 6-1 which deal with the more technical aspects of the research. However, addressing high volumes of data is also done by examining the more methodological aspects as well, such as 1, 6, 10 and 11. These aspects and how they relate to coding theme of high volumes of data is listed in Table 7- 6 below.

1	*Cyclical or iterative methodologies allow requirements to be added in new iterations*. When developing cyclically, the volume of data in the data warehouse does not suddenly pose a problem after an initial load. Data will be added to the warehouse incrementally, thereby allowing the team more time to monitor and plan for expanding capacity and improving performance.
4	*Priority of preventive maintenance and when it is performed.* This research aspect relates to the management attitude toward maintenance. As high volumes of data is often something that is only experienced on mature systems, managers often tend to neglect preventive maintenance until it becomes a problem.
6	*Use of dimensional models.* The use of conformed dimensions and the Kimball bus matrix assists in preventing duplicate data and planning for expansion of the data warehouse.
7	*Addressing challenges of high volumes of data.* Challenges surrounding high volumes of data is prevalent in the literature. Addressing these challenges often requires addressing management attitude toward preventive maintenance.
8	*Partitioning scheme usage and strategy.* This research aspect arose from Inmon's recommendation on logical partitioning data according to data which is accessed frequently and data which is less frequently required.
10	*Is maintenance being done primarily by junior developers?* Preventive maintenance can be difficult as performance tuning and capacity planning require a broad spectrum of knowledge.
11	*Use of outsourcing.* Outsourced developers have a specific set of work to perform. This is often only what is included in the outsourcing contract and the time spent performing that task is limited. As such, when outsourcing is used to accomplish preventive or perfective maintenance, it is possible that the solution will be very specific and short lived. High volumes is an ongoing problem which requires constant monitoring and tweaking.
13	*Logical hardware separation*. Inmon recommends that data be separated logically into sectors. For example, The near line sector which lies on more costly hardware is reserved for data recently pulled through from transactional systems that is reported on frequently and the archival sector which resides on less costly hardware contains data that is only used on historical reports.

Table 7-6 Research aspects related to coding theme "high volumes"

The challenge of high volumes of data in a data warehouse is a common one. The researcher concludes that areas to be investigated further in addressing this challenge are implementation of partitioning strategies, identifying and reducing duplicate data and creating data storage and access strategies according to these partitioning strategies. These solutions may treat the symptoms and not the cause. The root cause of this challenge may be a lack of understanding by executives and

managers who tend to put emphasis on delivering a product. This is why many participants perform preventive maintenance and experience the challenge of high volumes of data. Only when usage increases and data throughput increases, does the volume of data pose a problem.

7.4.7 Theme 7: Metadata

Metadata is a subject that many participants of the interpretive research questionnaire were unfamiliar with. The research portion of this topic is reported in Section 6.6.7.

Metadata is often consumed by development and maintenance tools. In Section 6.6.13, it was observed that the participants do not use such tools, which indicates another area that metadata is being overlooked.

Metadata is shown to be a topic clearly considered in data warehouse literature which was reported in Chapter 4. Metadata can be used to quickly generate documentation and is also useful when performing maintenance tasks as the understanding of metadata of the data warehouse can often lead to solving a maintenance problem quickly.

Metadata is used to keep the data warehouse and the source systems in synchronisation. When a change to a remotely managed source system occurs, it has the potential to cause data loads to fail. If a metadata update or refresh occurs, the source system changes can be automatically detected and catered for.

Maintenance tools have the capability to use metadata to generate documentation and perform automated maintenance tasks.

The disregard of metadata amongst the participants leads the researcher to conclude that there is a lack of theoretical knowledge on metadata. A Kimball data warehouse is defined as being metadata driven hence it should be imperative that data warehouse professionals make use of it. The disregard of metadata is practice redolent of regular transactional system developers.

7.4.8 Theme 8: Methodology

Questions were posed to participants to determine if they used a cyclical or waterfall type of methodology. Some participants indicated that they were following the Kimball life cycle approach which ought to imply a cyclical approach however this was not the case. The code “Cyclical development not being practised” was coded in 6 participants’ responses.

The first research aspect of Table 6-1 relates to methodology directly. Cyclical methodologies are recommended for data warehouse development as requirements frequently change.

The topic of methodology is linked to almost all aspects from the literature review as well as almost all themes from the research discussion. From the responses received from participants, the researcher concludes that there is a lack of theoretical knowledge of methodologies in data warehouse professionals in South Africa.

7.4.9 Theme 9: OLTP comparison

In the interpretive study, questions were posed to participants about their experience with OLTP systems in order to understand the participants’ opinion of the difference between OLTP development and maintenance and DW development and maintenance. Unfortunately, as the participants were sourced for their experience in

data warehouse systems, there weren't many who had OLTP experience. It was only P7 and P11 that had meaningful insight.

The literature surveyed was predominantly about data warehouse systems. There were many aspects of research that overlap with OLTP systems. These include methodology, maintenance approach, outsourcing and general attitude. While these aspects overlap, there are differences in the way these aspects are handled for OLTP systems and the way they are handled for data warehouse systems. These differences lead to different challenges.

The literature suggests that OLTP systems are generally not used out of choice but out of necessity. Hence, it is often an IT department driving the change and users have no choice but to learn a new system. Data warehouse and business intelligence systems are more often used out of choice as a user is trying to analyse some information. Hence, requests for changes and enhancements originate from the user. Thus, it is more challenging to make a successful DW system as users may choose not to use it

Even though the participants did not have experience with transactional systems, their responses indicated that many of the development and maintenance methods that were being practised were more aligned with those of transactional systems than with that of data warehouse systems. It is therefore concluded by the researcher that the higher level development staffs such as the program manager or development manager have not carefully considered the differences in methodology between data warehouse systems and transactional systems before embarking of the building of a data warehouse.

7.4.10 Theme 10: Outsourcing

The responses and coding on this theme, discussed in Section 6.6.9 revealed a negative view toward outsourcing from the participants which was expected from the literature viewed.

Aspect 11, “Use of outsourcing” was identified as an empirical research aspect from the literature to include in the interpretive questionnaire. Table 7-7 below contains the other aspects that relate to the use of outsourcing and a description of this relationship.

1	*Cyclical or iterative methodologies allow requirements to be added in new iterations.* Methodology relates to outsourcing because outsourcing contracts have specific goals and timelines. With cyclical development, requirements may shift due to shifting priorities. It is therefore more difficult to use outsourced developers when doing cyclical development.
2	*Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.* The use of outsourcing entails completing a project and handing over to an internal team for maintenance or conversely, handing maintenance over to an outsourced team. Both of these negatively affect maintenance of a DW project.
9	*Training of users and cross skilling of developers.* It is not in the interest of the outsourcing company to participate in cross-skilling of developers outside of their company. This would jeopardize future business prospects.
11	*Use of outsourcing.* There are many warnings present in the literature about managing outsourced teams.
12	*Involvement of development team on maintenance tasks*. This is similar to aspect 2. Even if the developed team is outsourced, they should be involved in continual maintenance of the system.

Table 7-7 Research aspects related to coding theme “outsourcing”

Despite the negative view of outsourcing from DW professionals, outsourcing is being used in data warehouse development and maintenance as broadly as it being used in transactional system development and maintenance. The researcher concludes that this is either out of necessity due to lack of available skilled resources or it may be that managers responsible for staffing a project do not have the understanding of data warehouse methodology. In order to make outsourcing more successful, clear guidelines and contracts should be drawn up and a handover should be included.

7.4.11 Theme 11: Preventive/perfective maintenance

Section 6.6.11 contains a discussion on the research analysis portion of preventive maintenance. It is observed here that preventive maintenance is an ongoing problem and while there is effort made to carry out preventive or perfective maintenance, it is not enough to curb the challenge of high volumes or long data load times.

The theme of preventive and perfective maintenance is prominent in the literature being surveyed. A lot of emphasis is placed on the priority that preventive maintenance takes. Table 7-8 contains the aspects related to Preventive/Perfective maintenance

4	*Priority of preventive maintenance and when it is performed.* Ongoing preventive maintenance should be common practice on a DW system. This corresponds with the cyclical development methodology. Systems built using a sequential methodology such as the waterfall methodology would suffer from *ad hoc* preventive maintenance.
7	*Addressing challenges of high volumes of data.* Addressing challenges associated with high volumes of data involves performing and preventive perfective maintenance. If this type of maintenance is done pro-actively, then there would not be a need to perform emergency perfective maintenance which might be costly due to system downtime.
8	*Partitioning scheme usage and strategy.* Designing with a partitioning scheme in mind relates to the theme of preventive/perfective maintenance as this type of design is done for performance.
9	*Training of users and cross skilling of developers.* Training of users and developers relates to preventive maintenance because trained users would log less support requests and cross-skilling of developers aids quicker and more efficient maintenance times.
Aspects 10, 11, 13 relate to perfective maintenance in the same way as they relate to the theme of “High Volumes”. These are repeated here for readability.	
10	*Is maintenance being done primarily by junior developers?* Preventive maintenance can be difficult as performance tuning and capacity planning require a broad spectrum of knowledge.
11	*Use of outsourcing.* Outsourced developers have a specific set of work to perform. This is often only what is included in the outsourcing contract and the time spent performing that task is limited. As such, when outsourcing is used to accomplish preventive or perfective maintenance, it is possible that the solution will be very specific and short lived. High volumes is an ongoing problem which requires constant monitoring and tweaking.
13	*Logical hardware separation.* Inmon recommends that data be separated logically into sectors. For example, The near line sector which lies on more costly hardware is reserved for data recently pulled through from transactional systems that is reported on frequently and the archival sector which resides on less costly hardware contains data that is only used on historical reports.

Table 7-8 Research aspects related to coding theme “preventive/perfective maintenance”

On the aspect of priority of preventive maintenance, much of the literature on transactional systems corresponds with the experience of developers on data warehouse systems. It is expressed in the literature reviewed in Section 5.3 on preventive maintenance that preventive maintenance does not enjoy priority in transactional systems.

7.4.12 Theme 12: Training

From the interpretive study, it was indicated in Section 6.6.12 that user interaction with developers is uncommon and discouraged. This demonstrates a closer alignment to transactional system development and maintenance principles.

Research aspect 9 was identified as "Training of users and cross skilling of developers." Training is a research aspect which can be related to many others. Table 7-9 lists these relationships.

1	*Cyclical or iterative methodologies allow requirements to be added in new iterations.* Cyclical methodologies will allow for smaller chunks of a system being released, lessening the amount of learning that users are required to undertake.
2	*Question of whether original developers stay on the DW project or if it is handed over to another team for maintenance.* More training is required amongst developers if the maintenance is to be taken over by another department.
3	*Use of a steering committee (which includes representation from the source systems, business management and IT)*. The steering committee would be responsible for ensuring that training is budgeted for and conducted for all required users.
5	*Use of metadata including documentation*. Formal training requires training manuals. These are easier to create if documentation is in place or can be generated using metadata.
6	*Use of dimensional models.* Dimensional models are easy to understand and recognizable. The same model can be applied to different business processes making it easier to understand the data.
10	*Is maintenance being done primarily by junior developers?* Junior developers require extra time to learn the technology being used as well as the DW system. If junior developers are assisting with user support, it may be more frustrating for users.
11	*Use of outsourcing.* If a system is developed by an outsourced company, training of users and maintenance developers might require back and forth communication with this company.
12	*Involvement of development team on maintenance tasks.* If the development team is involved in maintenance, it will be easier to train fellow developers and users.

Table 7-9 Research aspects related to coding theme "training"

User training is part of the deployment phase of the Kimball life cycle as discussed in Section 4.3.2.8. It is recommended that training workshops be set up as part of a marketing campaign to attract users to use the new DW system. In contrast to this, users are forced to use a transactional system once it is set up and training is usually mandatory.

User interaction with developers is generally kept to a minimum on a transactional system. This could be due to developers not understanding the requirements from a business point of view. Hence users tend to interact only with business analysts. Interaction between users and developers is encouraged in a data warehouse environment. As discussed in Section 4.3, this interaction is mutually beneficial. Users are supported by developers who in turn better understand requirements for the next phase of development.

The researcher concludes that the reason for the lack of alignment between literature and the practice by DW professionals is that the cost of developer resources are too high for them to spend time supporting users. Another possibility is that developers do not spend enough time understanding the business processes and terminology for the interaction to be productive.

7.4.13 Theme 13: Use of tools

Research aspect 13, “Usage and availability of maintenance tools” was identified in the literature as a possible means to assist with maintenance tasks. These may have included metadata tools, or automated coding tools. The research aspect guided some questions on development tools. Unfortunately, the bulk of the participants indicated that they were not being used.

The literature on the topic of software tools geared specifically toward maintenance is also limited. However, many development tools can also be used for maintenance.

These tools use metadata to provide the developer with useful information such as data links to source systems, data type information, software version differences and code summary information. This type of information can speed up maintenance as the developer will no longer need to search through documentation which may not be current.

The researcher concludes that maintenance tools should be further investigated as the use of tools has the potential to assist in maintenance. Lack of knowledge of these tools is related to lack of knowledge of metadata and the uses of metadata from a developer perspective. In transactional development, the use of metadata is not emphasised which has also influenced its use on data warehouse maintenance in a South African context.

7.5 Final conclusions of study

The diagram at the beginning of this chapter shows the path taken during this dissertation to reach the final conclusions.

In Chapter 3, literature on transactional development and maintenance was reviewed. Transactional systems development and maintenance methodologies have been practised for a longer time than DW methodologies. Software development and maintenance professionals, including those specialising in data warehousing are very familiar with these methods. The methodology followed during software development and maintenance is sequential which is not suitable to DW development and maintenance.

In Chapter 4, literature on DW development and maintenance was reviewed. These methodologies are less established than transactional software development and maintenance methodologies. As a result, some DW professionals are not wholly familiar with DW methodologies and follow a more familiar sequential transactional development and maintenance methodology instead.

In this chapter, the research aspects acquired in Chapter 5 were compared with the research themes acquired in Chapter 6. The conclusions drawn are that many DW professionals are not utilising the benefits of a DW methodology when developing and maintaining data warehouses but rather using a sequential methodology more closely associated with transactional system development and maintenance. This leads to challenges during DW development and maintenance which are not experienced during transactional system development and maintenance.

The challenges of preventive maintenance experienced by researchers align with challenges reported in literature in Chapter 5. This is expected when developing a DW using a sequential process due to the phased nature of transactional system development. Hence the conclusion reached by the researcher is that data warehouse system development should be cyclical in order to allow preventive maintenance to form part of each cycle of development. Further to this, preventive maintenance is included in the planning process for data warehouses by Kimball *et al.* (2008: 381) as discussed in Section 4.3.2.9 and perfective maintenance forms a significant portion of the data warehouse development process in Inmon *et al.'s* (2008:168) DW 2.0. This level of attention is not awarded to preventive and perfective maintenance in any of the transactional system methodologies reviewed in Chapter 3.

Preventive and perfective maintenance should be a priority within the DW team. When prioritised, then this type of maintenance will occur in a proactive manner. To perform preventive maintenance in a reactive fashion is counter-productive as this might result in downtime or other types of expenses in the form of emergency maintenance.

It was found from the research conducted in Chapter 6 that steering committees are not being used for DW projects. A data warehouse development project should include a steering committee that is involved with the DW project. This committee would be responsible for budgeting and resource allocation approval. The committee

should encourage and necessitate training of users and communication between developers and the business. The use of a steering committee with representation from all departments within an enterprise also facilitates the admonishment of stove-pipe type of systems per department, allowing the data warehouse to grow.

It was also found that metadata is not well understood by DW professionals. This may be due to a lack of theoretical training in the South African industry. Due to this lack of understanding, metadata and DW development tools that harness the power of metadata is not being actively used by DW professionals. As a result, maintenance takes longer than necessary and there is a greater dependency on documentation.

In Chapter 6, it was discussed that there is a generally negative view toward outsourcing DW development and/or maintenance. Outsourcing can exacerbate communication difficulties, however outsourcing remains a viable solution. This practice also stems from outsourcing being a common occurrence in transactional system development whereas it is warned against in DW literature as discussed in Chapter 4. The lack of use of a steering committee is related to this issue. A steering committee gathering to agree on DW program entails agreement from high level managers and executives across an organisation. In order for the data warehouse project to proceed, the steering committee would agree on a long term DW project commitment. As outsourcing is more beneficial in the short term, agreement on time and budget for a long term project would allow for a DW team to be recruited for internal development and maintenance.

7.6 **Limitations of study**

Due to the researcher being based in the US during a portion of the research duration, some geographical limitations were incurred. These led to some interviews being conducted telephonically and some participants opting to answer the interpretive research questionnaire in a written format.

There is also a limitation of saturation of analysis. Elliot *et al.* (1999: 222) outline that saturation occurs when understanding and consistency is achieved despite of unique responses. This is not always possible. Smith (1999: 409) debates that analysis is iterative hence it is possible that analysis of a current unit could lead to insight which results in re-analysing previous units.

Saturation is easier achieved when there is one research setting and participants share a common environment. In this case, participants had experience from different organisations however a level of generalisation can be achieved since many similar responses were given. Therefore, information can be acquired from the data that was analysed.

In this study, saturation was not been achieved. This is due to the limited number of responses to all questions in the interpretive questionnaire.

7.7 Further study considerations

As mentioned in Section 7.5 above saturation has not been achieved with this study. It would be beneficial to consider adding more participants so that a broader range of answers can be considered.

Most participants did not respond to the question on a comparison with transactional system development hence the comparison was done by comparing DW industry experience with the literature reviewed on transactional system development and adding the researchers own view to the topic being discussed. This dissertation has raised the question of whether data warehouse development is being influenced by transactional development principles and if so, should that be changed.

Many developers found documentation to be unhelpful or too verbose. This raises the question of the usefulness of data warehouse documentation. A study could be conducted comparing maintenance of a system by developers who are given current

documentation, developers who are given no documentation and developers who are given out-dated documentation.

While many of the participants may have a dissatisfied view of the process they follow, they may feel powerless to change it. Questions this raise are: how does a methodology change come about? Which role brings about this change?

There have been many comparisons on methodology from a development perspective. How do these methodologies compare from the perspective of user satisfaction, user support, ease of maintenance and long term return on investment?

7.8 Chapter summary

The aims and objectives of this dissertation were recollected in Section 7.2

An evaluation of interpretive research performed was arranged in Section 7.3 according to criterion suggested by Klein & Myers.

In Section 7.4 each research theme discovered during the coding process is linked to research aspects on which the interpretive research questionnaire was based. In Section 7.5 these links are used to present final conclusions on the study.

Limitations of the study were presented in Section 7.6 and further study considerations were suggested in Section 7.7.

This chapter concludes the dissertation.

References

Adler, P., & Adler, P. 1994. Observational Techniques. *Handbook of Qualitative Research*: 377–392.

Agrawal, D., El Abbadi, A., Singh, A., & Yurek, T. 1997. Efficient view maintenance at data warehouses. *ACM SIGMOD Record*: 417–427. http://dl.acm.org/citation.cfm?id=253355, November 14, 2012.

Ahmed, R. E. 2006. Software maintenance outsourcing: Issues and strategies. *Computers & Electrical Engineering*, 32(6): 449–453.

Ahrens, T., & Chapman, C. S. 2006. Doing qualitative field research in management accounting: Positioning data to contribute to theory. *Accounting, Organizations and Society*, 31(8): 819–841.

Anquetil, N., de Oliveira, K. M., de Sousa, K. D., & Batista Dias, M. G. 2007. Software maintenance seen as a knowledge management issue. *Information and Software Technology*, 49(5): 515–529.

Bacharach, S. B. 1989. Organizational theories: Some criteria for evaluation. *Academy of management review*, 496–515.

Bailin, S. C. 1989. An object-oriented requirements specifications method. *Communications of the ACM*, 32(5): 608–623.

Basili, V. R., & Boehm, B. 2001. COTS-based systems top 10 list. *Computer*, 34(5): 91–95.

Baskerville, R. L. 1999. Investigating information systems with action research. *Communications of the AIS*, 2(3es): 4.

Baskerville, R., & Myers, M. D. 2004. Special issue on action research in information systems: Making IS research relevant to practice. *MIS Quarterly*, 28(3): 329–335.

Begel, A., & Nagappan, N. 2007. Usage and perceptions of agile software development in an industrial context: An exploratory study. *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*: 255–264. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4343753, March 11, 2013.

Bennett, K. H. 1995. Legacy systems: coping with stress. *Software, IEEE*, 12(1): 19–23.

Bennett, K. H., & Rajlich, V. T. 2000. Software maintenance and evolution: a roadmap. *Proceedings of the conference on The future of Software engineering*: 87.

Bhatti, M. W., Hayat, F., Ehsan, N., Ishaque, A., Ahmed, S., & Mirza, E. 2010. A methodology to manage the changing requirements of a software project: 319–322. Presented at the International Conference on Computer Information Systems and Industrial Management Applications (CISIM), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5643642, July 22, 2012, IEEE.

Bhatt, P., Shroff, G., & Misra, A. K. 2004. Dynamics of software maintenance. *ACM SIGSOFT Software Engineering Notes*, 29(5): 1–5.

Bhatt, P., Shroff, G., & Misra, A. K. 2006. Influencing factors in outsourced software maintenance. *ACM SIGSOFT Software Engineering Notes*, 31(3): 1–6.

Boehm, B. 2002. Get ready for agile methods, with care. *Computer*, 35(1): 64–69.

Boehm, B. 2006. A view of 20th and 21st century software engineering. *Proceedings of the 28th international conference on Software engineering*: 12–29.

Boehm, B. W. 1988. A spiral model of software development and enhancement. *Computer*, 21(5): 61–72.

Boehm, B. W., Gray, T. E., & Seewaldt, T. 1984. Prototyping versus specifying: a multiproject experiment. *Software Engineering, IEEE Transactions on*, (3): 290–303.

Boehm, B. W., & Papaccio, P. N. 1988. Understanding and controlling software costs. *Software Engineering, IEEE Transactions on*, 14(10): 1462–1477.

Booch, G. 1994. *Object-oriented analysis and design with applications* (2nd ed.). Addison Wesley Longman, Inc.

Braude, E., & Bernstein, M. 2011. *Software Engineering: Modern Approaches* (2nd ed.). John Wiley & Sons, Inc.

Brooks, N. 2006. Understanding IT outsourcing and its potential effects on IT workers and their environment. *Journal of Computer Information Systems*.

Budd, T. 1997. *Object-Orented Programming* (Second Edition.).

Burck, C. 2005. Comparing qualitative research methodologies for systemic research: The use of grounded theory, discourse analysis and narrative analysis. *Journal of Family Therapy*, 27(3): 237–262.

Burns, R. 2000. *Introduction to Research Methods*. Sage Publications.

Campbell, J. D. 1995. Outsourcing in maintenance management: a valid alternative to self-provision. *Journal of Quality in Maintenance Engineering*, 1(3): 18–24.

Chapin, N. 2000. Do we know what preventive maintenance is? *Software Maintenance, 2000. Proceedings. International Conference on*: 15–17.

Chaudhuri, S., Dayal, U., & Narasayya, V. 2011. An overview of business intelligence technology. *Communications of the ACM*, 54(8): 88–98.

Cockburn, A., & Highsmith, J. 2001. Agile software development, the people factor. *Computer*, 34(11): 131–133.

Cohen, D., & Crabtree, B. 2006. Semi-structured Interviews. *Qualitative Research Guidelines Project*. http://www.qualres.org/HomeSemi-3629.html, March 7, 2013.

Cohn, M. 2004. *User Stories Applied for Agile Software Development*. Addison Wesley.

Cohn, M. 2008. Getting Agile with Scrum. Presented at the SD Best Practices Conferences & Expo, http://www.mountaingoatsoftware.com/presentations/91-getting-agile-with-scrum, January 4, 2012.

Cohn, M. 2011a. Introduction to Scrum - An Agile Process. http://www.mountaingoatsoftware.com/topics/scrum, December 14, 2011.

Cohn, M. 2011b. Agile Training on Sprint Planning Meeting. http://www.mountaingoatsoftware.com/scrum/sprint-planning-meeting, December 31, 2011.

Cohn, M. 2011c. Training for Scrum Task Board Use. http://www.mountaingoatsoftware.com/scrum/task-boards, December 31, 2011.

Cohn, M. 2011d. Sprint Review Meeting. http://www.mountaingoatsoftware.com/scrum/sprint-review-meeting, January 4, 2012.

Corbin, J., & Strauss, A. 1990. Grounded theory method: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 3–21.

Cordy, J. 2003. Comprehending Reality - Practical Barriers to Industrial Adoption of Software Maintenance Automation. *Proceedings of the 11 th IEEE International Workshop on Program Comprehension*. Presented at the 11 th IEEE International Workshop on Program Comprehension.

Costa, R. 2011. Information Systems and Qualitative Research, by Raul Silva Costa. http://www.map.edu.pt/i/2008/map-i-research-methods-workshop-2009/RaulCostaFinal.pdf/view, May 29, 2011.

Creswell, J. 2003. *Research Design: Qualitative, Quantitative and Mixed Methods Approaches* (2nd ed.). Sage Publications.

Crotty, M. 1998. *The Foundations of Social Research: meaning and perspective in the research process*. Sage Publications.

DATA WAREHOUSE NETWORK. 1998. . http://indigo.ie/~dataware/, February 19, 2011.

Dayal, U., Castellanos, M., Simitsis, A., & Wilkinson, K. 2009. Data integration flows for business intelligence. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*: 1–11. http://dl.acm.org/citation.cfm?id=1516362, July 19, 2012.

Dedeke, A. N. 2012. Improving Legacy-System Sustainability: A Systematic Approach. *IT Professional*, 14(1): 38–43.

Dekleva, S. M. 1992. The influence of the information systems development approach on maintenance. *MIS Quarterly*, 16(3): 355–372.

De Villiers, M. R. 2005. Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*: 142–151.

Dishaw, M. T., & Strong, D. M. 1998. Assessing software maintenance tool utilization using task–technology fit and fitness-for-use models. *Journal of Software Maintenance: Research and Practice*, 10(3): 151–179.

Do, L., Drew, P., Jin, W., Jumani, V., & Van Rossum, D. 1998. Issues in developing very large data warehouses. *PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES*: 633–636. http://www.vldb.org/conf/1998/p633.pdf, February 7, 2013.

Driscoll, D. L., Appiah-Yeboah, A., Salib, P., & Rupert, D. J. 2007. Merging qualitative and quantitative data in mixed methods research: How to and why not. *Ecological and Environmental Anthropology (University of Georgia)*, 18.

Dubin, R. 1978. *Theory building* (Rev. ed.). New York: Free Press.

Elliot, R., Fischer, C. T., & Rennie, D. L. 1999. Evolving guidelines for the publication of qualitative research studies in psychology and related fields. *British Journal of Clinical Psychology*, 38: 215–229.

English, L. 2004. Six Sigma and Total Information Quality Management - Information Management Magazine Article. http://www.information-management.com/issues/20041001/1011016-1.html?zkPrintable=1&nopagination=1, February 19, 2013.

Fichman, R. G., & Kemerer, C. F. 1992. Object-oriented and conventional analysis and design methodologies. *Computer*, 25(10): 22–39.

Fontana, A., & Frey, J. 1994. Interviewing: The art of science. *Handbook of Qualitative Research*: 361–376.

Fossey, E., Harvey, C., McDermott, F., & Davidson, L. 2002. Understanding and evaluating qualitative research. *Australian and New Zealand journal of psychiatry*, 36(6): 717–732.

Friesen, N. 2008. Critical Theory: Ideology Critique and the myths of e-learning. *Ubiquity*, 2008(June). http://ubiquity.acm.org.nwulib.nwu.ac.za/article.cfm?id=1386860, May 29, 2011.

García, I., Pacheco, C., Mendoza, E., Calvo-Manzano, J. A., Cuevas, G., & San Feliu, T. 2010. Managing the software process with a software process improvement tool in a small enterprise. *Journal of Software Maintenance and Evolution: Research and Practice*, n/a–n/a.

Glaser, B. 2008. Conceptualization: On theory and theorizing using grounded theory. *International Journal of Qualitative Methods*, 1(2): 23–38.

Glaser, B., & Strauss, A. 1967. *The Discovery of Grounded Theory. Strategies for Qualitative Research*. AldineTransaction.

Glaser, & Holten. 2004. Remodeling grounded theory. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*. http://www.qualitative-research.net/ fqstexte/2-04/2-04glaser-e.htm.

Goubil-Gambrell, P. 1991. What do practitioners need to know about research methodology? *Professional Communication Conference, 1991. IPCC'91. Proceedings. The Engineered Communication., International*: 243–248.

Graneheim, U. H., & Lundman, B. 2004. Qualitative content analysis in nursing research: concepts, procedures and measures to achieve trustworthiness. *Nurse education today*, 24(2): 105–112.

Hammersley, M., & Atkinson., P. 2007. *Ethnography:principles in practice* (3rd ed.). Taylor & Francis e-Library.

Harker, S. D. P., Eason, K. D., & Dobson, J. E. 1993. The change and evolution of requirements as a challenge to the practice of software engineering: 266–272.

Presented at the Proceedings of IEEE International Symposium onRequirements Engineering, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=324847, July 22, 2012, IEEE Comput. Soc. Press.

Hayata, T., & Han, J. 2011. A hybrid model for IT project with Scrum. *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on*: 285–290.

Hayes, S., & Andrews, M. 2001. An introduction to agile methods. *Steve Hayes (Khatovar Technology) steve@ khatovartech. com http://www. khatovartech. com*.

Henderson-Sellers, B., & Edwards, J. M. 1990. The object-oriented systems life cycle. *Communications of the ACM*, 33(9): 142–159.

Highsmith, J., & Cockburn, A. 2001. Agile software development: The business of innovation. *Computer*, 34(9): 120–127.

Hsieh, H.-F., & Shannon, S. 2005. Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, 15(9): 1277–1288.

Hudak, P., & Jones, M. P. 1994. *Haskell vs. Ada vs. C++ vs. awk vs.... an experiment in software prototyping productivity*. Research Report No. YALEU/DCS/RR-1049. http://haskell.cs.yale.edu/?post_type=publication&p=366, New Haven, CT: Department of Computer Science, Yale University.

Hughes, J., King, V., Rodden, T., & Andersen, H. 1994. Moving out from the control room: ethnography in system design. *Proceedings of the 1994 ACM conference on Computer supported cooperative work*: 429–439.

Huo, M., Verner, J., Zhu, L., & Babar, M. A. 2004. Software quality and agile methods. *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*.

IEEE Std 610.121990. 1990. IEEE standard glossary of Software Engineering Terminology. *The Institute of Electrical and Electronics Engineers*.

IEEE Std 1219. 2003. IEEE Standard for software maintenance.

Iivari, J. 1996. Why are CASE tools not used? *Communications of the ACM*, 39(10): 94–103.

Inmon, W. H. 1992. *Building the Data Warehouse*. New York, NY, USA: John Wiley & Sons, Inc.

Inmon, W. H. 2002. *Building the Data Warehouse* (3rd ed.). New York, NY, USA: John Wiley & Sons, Inc.

Inmon, W. H. 2005. *Building the Data Warehouse*. New York, NY, USA: John Wiley & Sons, Inc.

Inmon, W. H. 2007. Bill Gates is a failure. *Corporate Information Factory (CIF) Resources by Bill Inmon, Inmon Data Systems*. http://www.inmoncif.com/view/20, February 18, 2011.

Inmon, W. H., Imhoff, C., & Sousa, R. 2001. *Corporate Information Factory* (2nd ed.). John Wiley & Sons Inc.

Inmon, W. H., Strauss, D., & Neushloss, G. 2008. *DW 2.0: The Architecture for the Next Generation of Data Warehousing*. Morgan Kaufmann Publishers.

Iqbal, R., Gatward, R., & James, A. 2005. A General Approach to Ethnographic Analysis for Systems Design. *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*: 235–238. Presented at the SIGDOC '05.

ISO/IEC 12207. 2007. Systems and software engineering — Software life cycle processes. *ISO, Geneva, Switzerland*.

ISO/IEC 14764. 2006. Software Engineering — Software Life Cycle Processes — Maintenance ISO/IEC 14764 IEEE Std 14764.

Jacsó, P. 2005. Google Scholar: the pros and the cons. *Online Information Review*, 29(2): 208–214.

Johnson, R. B., & Onwuegbuzie, A. J. 2004. Mixed methods research: A research paradigm whose time has come. *Educational researcher*, 33(7): 14.

Kajko-Mattsson, M. 2000. Preventive Maintenance! Do we know what it is? *Software Maintenance, 2000. Proceedings. International Conference on*: 12–14.

Kaplan, B., & Maxwell, J. 2005a. Qualitative research methods for evaluating computer information systems. *Evaluating the Organizational Impact of Healthcare Information Systems*, 30–55.

Kaplan, B., & Maxwell, J. 2005b. Qualitative research methods for evaluating computer information systems. *Evaluating the Organizational Impact of Healthcare Information Systems*, 30–55.

Kimball, R. 1996. *The data warehouse toolkit : practical techniques for building dimensional data warehouses*. John Wiley & Sons.

Kimball, R., Ross, M., Thornthwaite, W., & Mundy, J. 1998. *The Data Warehouse Lifecycle Toolkit*. John Wiley & Sons.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. 2008. *The Data Warehouse Lifecycle Toolkit* (2nd ed.). John Wiley & Sons.

Kim, M., Sazawal, V., Notkin, D., & Murphy, G. 2005. An empirical study of code clone genealogies. *ACM SIGSOFT Software Engineering Notes*, 30(5): 187–196.

Kincheloe, J. L., & McLaren, P. L. 1994. Rethinking Critical Theory and Qualitative Research. *Handbook of Qualitative Research*. Sage Publications.

Kitchenham, B. A., Travassos, G. H., von Mayrhauser, A., Niessink, F., Schneidewind, N. F., Singer, J., et al. 1999. Towards an ontology of software maintenance. *Journal of Software Maintenance: Research and Practice*, 11(6): 365–389.

Klein, H., & Myers, M. 1999. A SET OF PRINCIPLES FOR CONDUCTING AND EVALUATING INTERPRETIVE FIELD STUDIES IN INFORMATION SYSTEMS. *MIS Quarterly*, 23(1): 67–94.

Kniberg, H. 2007. *Scrum and XP from the Trenches: Enterprise Software Development*. Lulu.com.

Ko, A. J., Aung, H. H., & Myers, B. A. 2005. Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintenance tasks. *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*: 126–135. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1553555, January 21, 2013.

Korson, T., & McGregor, J. D. 1990. Understanding object-oriented: a unifying paradigm. *Communications of the ACM*, 33(9): 40–60.

Kotony, G., & Hutchinson, J. 2005. Managing change in COTS-based Systems. *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05)*: 69 – 78. Presented at the International Conference on Software Maintenance (ICSM '05), IEEE Computer Society.

LaRossa, R. 2005. Grounded theory methods and qualitative family research. *Journal of Marriage and Family*, 67(4): 837–857.

Lee, A. 1989. A Scientific Methodology for MIS Case Studies. *MIS Quarterly*, 13(1): 33–50.

Lee, A. J., Nica, A., & Rundensteiner, E. A. 2002. The EVE approach: view synchronization in dynamic distributed environments. *Knowledge and Data Engineering, IEEE Transactions on*, 14(5): 931–954.

Legodi, I., & Barry, M. L. 2008. The current challenges and status of risk management in enterprise data warehouse projects in South Africa. *Technology Management for Global Economic Growth (PICMET), 2010 Proceedings of PICMET'10:* 1–5.

Lehman, M. M. 1980. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9): 1060–1076.

Lehman, M. M., & Ramil, J. F. 2001. Rules and tools for software evolution planning and management. *Annals of software engineering*, 11(1): 15–44.

Leonard, J., Pardoe, J., & Wade, S. 1988. Software maintenance-cinderella is still not getting to the ball. *Software Engineering, 1988 Software Engineering 88., Second IEE/BCS*

Conference: 104–106. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=196369, December 27, 2012.

Lethbridge, T. C., Singer, J., & Forward, A. 2003. How software engineers use documentation: The state of the practice. *Software, IEEE*, 20(6): 35–39.

Lewis, W. 2009. *Software testing and continuous quality improvement*. Auerbach Publications.

Lientz, B. P. 1983. Issues in software maintenance. *ACM Computing Surveys (CSUR)*, 15(3): 271–278.

Lientz, B. P., & Swanson, E. B. 1981. Problems in application software maintenance. *Communications of the ACM*, 24(11): 763–769.

List, B., Bruckner, R., Machaczek, K., & Schiefer, J. 2002. A comparison of data warehouse development methodologies case study of the process warehouse. *Database and Expert Systems Applications*: 203–215.

Lunan Li. 2010. A framework study of ETL processes optimization based on metadata repository: V6–125–V6–129. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5486338, July 21, 2012, IEEE.

Manchanda, S., Singh, S. B., & Dave, M. 2009. CHANGE MANAGEMENT AND SOFTWARE REUSE SUPPORTIVE “GENETIC INFORMATION SYSTEM DEVELOPMENT AND MAINTENANCE” MODEL, 19(1): 113 –136.

Marczyk, G., DeMatteo, D., & Festinger, D. 2005. *Essentials of Research Design and Methodology*. John Wiley & Sons, Inc.

Mayring, P. 2000. Qualitative content analysis. *Forum: Qualitative social research*: 2–00.

Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R., & Jazayeri, M. 2005. Challenges in software evolution. *Principles of Software Evolution, Eighth International Workshop on*: 13–22.

Microsoft Technet. 2000. Chapter 12 - Data Warehousing and Online Analytical Processing. *Microsoft SQL 70 Product Documentation*. http://technet.microsoft.com/en-us/library/cc917548.aspx, February 17, 2013.

Mookerjee, R. 2005. MAINTAINING ENTERPRISE SOFTWARE APPLICATIONS. *COMMUNICATIONS OF THE ACM*, 48(11): 75 – 79.

MSDN. 2010. Sprint Burndown. *Microsoft Developer Network*. http://msdn.microsoft.com/en-us/library/ff731588.aspx, January 2, 2012.

Myers, M. D. 1997. Qualitative Research in Information Systems. *MIS Quarterly*, 21(2): 241–242.

Myers, M. D. 1999. Investigating information systems with ethnographic research. *Communications of the AIS*, 2(4): 1.

Negash, S. 2004. Business Intelligence. *Communications of the Association for Information Systems*, 13: 177–195.

Nerson, J. M. 1992. Applying object-oriented analysis and design. *Communications of the ACM*, 35(9): 63–74.

Nguyen, V., Boehm, B., & Danphitsanuphan, P. 2010. A controlled experiment in assessing and estimating software maintenance tasks. *Information and Software Technology*.

Niederman, F., Brancheau, J. C., & Wetherbe, J. C. 1991. Information systems management issues for the 1990s. *MIS quarterly*, 475–500.

Onwuegbuzie, A. J., & Leech, N. L. 2004. Enhancing the interpretation of “significant” findings: The role of mixed methods research. *The Qualitative Report*, 9(4): 770–792.

Opler, A. 1966. New directions in software 1960-1966. *Proceedings of the IEEE*, 54(12): 1757–1763.

Orlikowski, W. J., & Baroudi, J. J. 1991. Studying information technology in organizations: Research approaches and assumptions. *Information systems research*, 2(1): 1–28.

Overhage, S., Schlauderer, S., Birkmeier, D., & Miller, J. 2011. On the Developer Adoption of Scrum: A New Acceptance Model for Agile Methodologies: 1–10. Presented at the System Sciences (HICSS), 2011 44th Hawaii International Conference on.

Pather, S., & Remenyi, D. 2004. Some of the Philosophical Issues Underpinning Research in Information Systems: From Positivism to Critical Realism. *Proceedings of SAICSIT*, 141 – 146.

Pfleeger, S. L. 1998. The nature of system change [software]. *Software, IEEE*, 15(3): 87–90.

Polo, M., Piattini, M., Ruiz, F., & Calero, C. 1999. MANTEMA: A software maintenance methodology based on the ISO/IEC 12207 standard. *Software Engineering Standards, 1999. Proceedings. Fourth IEEE International Symposium and Forum on*: 76–81.

Power, D. J. 2003. A brief history of decision support systems. *DSSResources. com*. http://dssresources.com/history/dsshistory.html.

Pressman, R. 2001. *Software Engineering: A practitioner's approach* (5th ed.). McGraw Hill.

QianCheng, Y. 2010. Metadata integration architecture in enterprise data warehouse system. *Information Science and Engineering (ICISE), 2010 2nd International Conference on*: 340–343. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5690320, July 19, 2012.

Ramanujan, S., Scamell, R. W., & Shah, J. R. 2000. An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort. *Journal of Systems and Software*, 54(2): 137–157.

Rao, N. M. 2009. Challenges in execution of outsourcing contracts. *Proceedings of the 2nd India software engineering conference*: 75–79.

Reifer, D. J., Basili, V. R., Boehm, B. W., & Clark, B. 2003. Eight lessons learned during COTS-based systems maintenance. *Software, IEEE*, 20(5): 94–96.

Ross, M. 2005. Slowly Changing Dimensions Are Not Always as Easy as 1, 2, 3. *http://www.kimballgroup.com*. http://www.kimballgroup.com/2005/03/slowly-changing-dimensions-are-not-always-as-easy-as-1-2-3/, October 19, 2014.

Ross, M. 2013. Design Tip #152 Slowly Changing Dimension Types 0, 4, 5, 6 and 7. *http://www.kimballgroup.com*. http://www.kimballgroup.com/2013/02/design-tip-152-slowly-changing-dimension-types-0-4-5-6-7/, October 19, 2014.

Rundensteiner, E. A., Koeller, A., & Zhang, X. 2000. Maintaining data warehouses over changing information sources. *Communications of the ACM*, 43(6): 57–62.

Salonen, J., & Pirttimäki, V. 2005. Outsourcing a Business Intelligence Function. *Frontiers of e-business research*, 661–675.

Schilling, J. 2006. On the Pragmatics of Qualitative Assessment. *European Journal of Psychological Assessment*, 22: 28–37.

Schneidewind, N. F. 1987. The State of Software Maintenance. *IEEE Transactions on Software Engineering*, SE-13(3): 303–310.

Schwaber, K., & Beedle, M. 2002. *Agile Software Development with Scrum*. Prentice Hall.

Seidewitz, E., & Stark, M. 1987. Toward a general object-oriented software development methodology. *ACM SIGAda Ada Letters*, III(4): 54–87.

Selic, B. 2009. Agile Documentation, Anyone? *Software, IEEE*, 26(6): 11–12.

Sen, A., & Sinha, A. P. 2005. A comparison of data warehousing methodologies. *Communications of the ACM*, 48(3): 79–84.

Shanks, G. 2007. Guidelines for conducting positivist case study research in information systems. *Australasian Journal of Information Systems*, 10(1).

Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., & Carlsson, C. 2002. Past, present, and future of decision support technology. *Decision support systems*, 33(2): 111–126.

Shin, B. 2003. An exploratory investigation of system success factors in data warehousing. *Journal of the Association for Information Systems*, 4: 141–170.

Singh, Y., & Goel, B. 2007. A step towards software preventive maintenance. *ACM SIGSOFT Software Engineering Notes*, 32(4): 10.

Smith, J. A. 1999. Towards a relational self: Social engagement during pregnancy and psychological preparation for motherhood. *British Journal of Social Psychology*, 38: 409 – 426.

Soni, N. 2007. Scrum Alliance - Keep Your Team Seeing RED. http://www.scrumalliance.org/articles/80-keep-your-team-seeing-red, December 18, 2011.

Sprague, R. H. 1980. A framework for the development of decision support systems. *MIS Quarterly*, 4(4): 1–26.

Stachour, P., & Collier-Brown, D. 2009. You don't know jack about software maintenance. *Communications of the ACM*, 52(11): 54–58.

Sulaiman, S., Idris, N. B., & Sahibuddin, S. 2002. Production and maintenance of system documentation: what, why, when and how tools should support the practice: 558–567. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1183081, March 15, 2012, IEEE Comput. Soc.

Susman, G. I., & Evered, R. D. 1978. An assessment of the scientific merits of action research. *Administrative science quarterly*, 582–603.

Sutherland, J., Downey, S., & Granvik, B. 2009. Shock Therapy: A Bootstrap for Hyper-Productive Scrum: 69–73. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5261105, December 15, 2011, IEEE.

Swanson, E. B. 1976. The dimensions of maintenance. *Proceedings of the 2nd international conference on Software engineering*: 492–497.

Thayer, R. H., & McGettrick, A. D. 2007. IEEE Software Engineering Standards: A Student's Version. *Software Engineering Education & Training, 2007. CSEET'07. 20th*

Conference on: 229–236. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4271610, October 25, 2012.

The Agile Alliance. 2001. Principles behind the Agile Manifesto. http://agilemanifesto.org/iso/en/principles.html, December 14, 2011.

Veerman, N. 2007. Automated mass maintenance of software assets. *Proceedings of the 11th European Conference on Software Maintenance and Reengineering*: 353–356.

VersionOne. 2010. State of Agile Development Survey Results - Agile Methods and Practices | VersionOne. *VersionOne - Agile made easier*. http://www.versionone.com/state_of_agile_development_survey/10/page3.asp, March 11, 2013.

Voas, J. 2007. A Baker's Dozen: 13 Software Engineering Challenges. *IT Pro*, (April): 48 – 53.

Von Mayrhauser, A., & Vans, A. M. 1995. Program comprehension during software maintenance and evolution. *Computer*, 28(8): 44–55.

Walizer, M. H., & Wienir, P. L. 1978. *Research Mthods and Analysis: Searching for Relationships*. Library of Congress Cataloging in Publication DAta.

Whitten, J., Bentley, L., & Dittman, K. 2004. *Systems Analysis and Design Methods* (6th ed.). McGraw Hill.

Widom, J. 1995. Research problems in data warehousing. *Proceedings of the fourth international conference on Information and knowledge management*: 25–30. http://dl.acm.org/citation.cfm?id=221319, November 12, 2012.

Xiong, C.-J., Xie, M., & Ng, S.-H. 2011. Optimal software maintenance policy considering unavailable time. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(1): 21–33.

Zhang, Y., & Wildemuth, B. M. 2009. Qualitative analysis of content. *Applications of social research methods to questions in information and library science*, 308–319.

Zhuge, Y., Garcia-Molina, H., Hammer, J., & Widom, J. 1995. View maintenance in a warehousing environment. *ACM SIGMOD Record*: 316–327. http://dl.acm.org/citation.cfm?id=223848, November 14, 2012.

Appendix A

Interview questionnaire

Interview introduction

Good day.

Thank you very much for taking your time to do this. The purpose of this is to assist me with conducting research for my master's dissertation at North West University (Potchefstroom campus).

The research is about trying to find out how data warehouse (DW) or business intelligence (BI) maintenance is done in practice hence there are no right or wrong answers – as the aim is to understand how things *are* done in real world situations – and not to examine how they should be done.

The interview is anonymous – your name will not appear in the research report. If this is telephonic, I would like to record the call so that I can take notes more accurately. Please let me know if this is acceptable. Feel free to ask me to remove anything you say or pause/stop the recording if you wish.

If you prefer to answer the questions without being interviewed, please feel free to record your answers and email them to me in an audio format. Alternatively, please type/write the answers to questions below and email to me at shakmj@gmail.com.

First I'm going to ask just a few short questions to understand the background of the interviewee – then I'm going to ask more open ended questions.

The open ended questions revolve around maintenance processes and practices. The questions are based on topics frequently covered in literature on software and data warehouse maintenance. Some of the topics are: standards, maintenance or change processes, methodology; as well as more practical topics surrounding documentation, learning, data volumes, etc. I hope to get descriptive answers to the longer questions so please give as much detail as you can. When answering, please try to remember all facets of your experience: people, processes and technical facets. Think of your experience with managers, clients, development teams. Think of the technical aspects such as the use of development tools, setting up of scheduled administrative tasks, and the use of documentation. Also, please think of the processes/standards that you follow.

If you do not understand a question, or feel that the question is not relevant to you, please state so and if possible, give a reason.

Please feel free to contact me if you have any follow-up questions or thoughts around these questions – even if they occur much later than we had communicated on the topic.

Again, thank you for your time and effort.

Warm Regards,

Shakeel Juggath

email: shakmj@gmail.com

skype: shak_sa

Interview Questions

Short questions:

1. Age:
2. Number of years experience in IT:
3. Number of years experience in software development:
4. Number of years experience in DW/BI:
5. Please list the DW/BI Technologies you have experience with and number of months/years experience:
6. Please list qualifications/certifications and year:
7. Please list any type of DW/BI training attended; whether at university, technikon, college, training vendor, technology specific training or other.

Descriptive questions:

1. Describe the methodology or development process that you go through when developing data warehouses and where maintenance fits into this methodology.
2. In particular, describe the process of making additions to the data warehouse after initial deployment.
3. Talk about some of the challenges in maintaining data warehouses/BI applications that you experienced.
4. How do you deal with handling and prioritization of ongoing requests
5. How do you address the challenge of working on a data warehouse that you have not worked on before?
6. Describe the use of documentation in maintaining and supporting data warehouses.

7. Describe when and how documentation is created.

8. Describe the relationship that DW/BI team has with business users and stakeholders and the interaction that goes on between the DW/BI team and the business users and stakeholders

9. How are the end users supported?

10. Describe the regular maintenance tasks that are performed on the data warehouse environment.

11. Are there maintenance tasks that you feel should be included and if so, what factors are influencing the implementation of these

12. Talk about your experience with the practical application of metadata in the maintenance of data warehouses and the other practical uses of metadata.

13. Describe the advantages or disadvantages of maintaining dimensional models.

14. Discuss what type (if any) monitoring, logging, administration, security auditing or usage logging that occurs on the DW/BI/ETL system

15. Describe the volumes data that you typically work and the types of challenges this poses?

16. How do you overcome the challenge of maintaining high volumes of data

17. Have you come across strategies for partitioning data – e.g. temporal data (which requires frequent updating) and relatively static data (which does not require frequent updating)

18. If you have experience as a consultant, talk about experience of DW/BI development/maintenance as being a consultant as opposed to being an in-house developer.

19. Tell us about your experience in Online-transactional processing (OLTP) systems development. In particular, describe the use of methodologies and how maintenance occurs.

20. How does OLTP differ from DW/BI in terms of development?

21. How does OLTP maintenance differ from DW/BI maintenance?

22. Are you using any industry standard or framework e.g. International Standard Organisation(ISO) standards for maintenance? If so, please describe the standard and the maintenance process.

23. From your perspective, what are the advantages and disadvantages of using consultants in DW/BI projects in terms of maintenance

24. How do development tools assist in maintenance? Have you used these in transactional maintenance or DW/BI maintenance?

25. Do you have experience with documentation tools in transactional or DW/BI maintenance? Can you explain how these work and if you find them useful?

Appendix B

Code List

The table below contains the code list per primary document. It lists all the codes created and the counts of the occurrence of the code per document.

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
1	*Ad hoc* additions	1	1	0	0	0	0	0	0	0	0	0	2
2	Adaptive maintenance treated as project	0	0	0	0	0	0	0	1	1	0	0	2
3	Agile type of methodology	0	1	1	0	0	0	0	1	0	1	0	4
4	Availability higher priority with OLTP	0	0	0	0	0	2	0	0	0	0	0	2
5	Business communication with team	0	0	0	0	0	0	0	0	0	0	0	0
6	Challenge of changing user mind-set to use DW	0	0	0	0	0	0	0	0	0	2	0	2
7	Challenge of data integrity	0	0	0	0	1	0	0	1	0	0	0	2
8	Challenge of duration for loading data	0	0	1	0	0	0	2	1	0	0	0	4
9	Challenge of DW methodology knowledge	0	0	0	0	0	0	1	0	0	0	0	1
10	Challenge of high volume addressed through perfective maintenance	0	0	0	0	0	1	1	0	0	0	0	2
11	Challenge of learning new system	0	0	0	0	0	0	1	0	0	0	0	1
12	Challenge of resource and time	0	0	0	0	1	0	0	0	0	0	0	1

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
	constraint												
13	Challenge of Team communication	0	0	1	0	0	0	2	0	1	0	0	4
14	Challenge of updating documentation	0	0	0	0	0	1	2	0	0	0	0	3
15	Challenges experienced	0	0	0	0	0	0	0	0	0	0	0	0
16	Communication challenges of outsourcing	4	0	0	0	0	2	1	0	1	0	0	8
17	Cyclical development	0	0	0	0	0	0	0	1	0	0	0	1
18	Cyclical development not being practised	2	2	1	1	0	1	0	0	0	0	1	8
19	Dedicated user support staff	0	0	0	0	0	0	0	0	1	0	0	1
20	Developer training	0	0	0	0	0	0	0	0	0	0	0	0
21	Development tools not well suited for maintenance	0	0	0	0	0	0	0	0	0	1	0	1
22	Different developers doing maintenance	1	0	0	0	0	0	1	0	0	0	0	2
23	Difficulty of using the technology	1	1	0	0	0	0	0	0	0	0	0	2
24	Dimensional models	0	0	0	0	0	0	0	0	0	0	0	0
25	Direct contact between business and team	0	0	2	0	2	0	0	0	0	0	0	4
26	Documentation availability	0	1	0	0	0	0	0	0	0	0	0	1
27	Documentation availability negative	1	1	1	0	1	0	1	0	0	0	0	5
28	Documentation availability positive	1	0	0	0	0	0	0	1	1	0	0	3
29	Documentation given lower priority	0	1	1	0	0	0	0	0	0	0	0	2
30	Documentation	0	0	0	0	0	0	0	0	0	0	0	0

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
	including use of metadata												
31	Documentation tool not useful	0	0	1	0	0	0	0	0	0	2	0	3
32	Documentation tool used	0	0	1	0	0	0	0	0	0	0	0	1
33	Documentation tools not being used	0	1	0	3	1	1	2	2	1	0	1	12
34	Documentation usefulness negative	0	0	2	0	0	1	2	0	0	2	1	8
35	Documentation usefulness positive	1	0	0	0	1	0	0	1	1	0	0	4
36	Does not make use of Kimball's bus matrix when planning changes	3	1	3	3	2	0	0	1	1	0	0	14
37	Does not use dimensional models	0	0	1	0	1	1	0	0	1	0	0	4
38	DW used to ease the load on transactional system	0	0	0	0	0	0	0	0	0	2	0	2
39	Effectiveness of Developer training	1	0	0	0	0	0	0	0	0	0	0	1
40	Experience with Preventive/Perfective maintenance	0	0	0	0	0	0	0	0	0	0	0	0
41	Hardware improvement to cope with volume	0	0	1	0	0	0	2	0	0	1	0	4
42	High cost of outsourcing	0	1	1	0	0	1	0	0	0	0	1	4
43	High volume require better user communication	0	0	0	0	0	0	1	0	1	0	1	3
44	High volumes pose challenge	0	1	2	2	1	2	3	2	1	1	0	15
45	High volumes require better design	0	1	0	2	1	0	1	2	0	1	0	8
46	Inmon type of	0	0	1	0	0	1	0	0	0	0	0	2

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
	philosophy												
47	Kimball methodology	1	0	0	1	0	0	0	1	0	1	0	4
48	Lack of communication between Developer and User	2	1	0	2	0	2	3	0	1	1	1	13
49	Lack of input from client	1	0	0	0	0	0	0	0	0	0	0	1
50	Lack of integration in team	1	0	0	0	0	0	0	0	0	0	0	1
51	Lack of preventive/perfective maintenance	1	2	1	0	1	0	3	0	0	0	0	8
52	Learning occurs under pressure	0	0	0	0	0	0	1	0	0	0	0	1
53	Lengthy documentation is counter productive	0	0	1	0	0	1	0	0	0	0	1	3
54	Low priority of perfective maintenance	1	0	0	0	0	1	1	0	0	2	0	5
55	Maintenance requires experienced professionals	0	0	0	0	0	0	1	0	1	0	0	2
56	Maintenance tools not being used	0	1	1	1	0	1	1	1	1	1	1	9
57	Management does not prioritise perfective maintenance	0	0	0	0	0	0	0	0	0	1	0	1
58	Materialized view problem	1	0	1	1	0	1	0	0	0	0	1	5
59	Metadata not understood by developer	1	0	0	0	0	1	0	0	0	0	0	2
60	Metadata not used by developer	1	1	1	1	0	1	0	0	0	1	0	6
61	Metadata not used	0	1	0	2	1	0	1	0	0	0	1	6

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
	for documentation												
62	Metadata use is manual	1	0	0	0	0	0	0	0	0	0	1	2
63	Metadata used by developer	0	0	0	0	1	0	0	0	0	0	1	2
64	Metadata used for documentation	0	0	1	0	0	0	0	0	0	0	0	1
65	Methodology comparison	0	0	0	0	0	0	0	0	0	0	0	0
66	Methodology linked to technology	0	1	0	0	0	0	0	0	0	0	0	1
67	More focus placed on perfective maintenance with OLTP	0	0	0	0	0	1	0	0	0	0	1	2
68	Multiple sources challenge	1	0	0	1	0	0	0	0	1	0	0	3
69	Not answered	0	0	3	5	1	1	4	1	0	0	4	19
70	Not familiar with Dimensional models	0	0	0	0	1	0	0	0	0	0	0	1
71	Not relevant	0	4	0	0	0	0	0	3	4	1	7	19
72	Oltp comparison	0	0	0	0	0	0	0	0	0	0	0	0
73	OLTP dev require more testing	0	0	0	0	0	0	0	0	0	1	0	1
74	OLTP forces users to change	0	0	0	0	0	0	0	0	1	0	0	1
75	OLTP given higher priority for maintenance	0	0	0	0	0	0	0	0	0	1	0	1
76	OLTP has simpler data retrieval	0	0	0	0	0	2	0	0	0	0	0	2
77	OLTP maintenance focus is on system architecture (hardware)	0	0	0	0	0	1	0	0	0	0	0	1
78	OLTP maintenance focuses on release	0	0	0	0	0	1	0	0	0	0	0	1

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
	management												
79	OLTP maintenance less challenging as data analysis not required	0	0	0	0	0	1	0	0	0	1	0	2
80	OLTP maintenance less challenging as reloads not required	0	0	0	0	0	1	0	0	0	1	0	2
81	Outsourced developers do more harm by focusing on getting the job done quickly	1	0	0	0	0	0	1	0	0	0	0	2
82	Outsourced developers have to spend time learning	1	0	1	1	0	0	3	1	1	0	0	8
83	Outsourcing	0	0	0	0	0	0	0	0	0	0	0	0
84	Outsourcing negative	1	1	0	1	0	0	3	0	1	0	0	7
85	Outsourcing pose risk of losing knowledge of a system	0	1	0	0	0	0	0	0	0	1	0	2
86	Outsourcing positive	0	0	2	0	0	0	0	0	0	0	1	3
87	Perfective maintenance is performed	0	0	1	2	2	2	0	2	1	0	1	11
88	Performance more important in OLTP	0	0	0	0	0	2	0	0	0	0	0	2
89	Preventive maintenance monitoring occurs	0	0	1	0	0	0	0	0	1	0	1	3
90	Preventive/Perfective maintenance	0	0	0	0	0	0	0	0	0	0	0	0
91	Priority based on deadlines	0	0	0	0	0	0	1	0	0	0	0	1
92	Priority on understanding	0	0	0	0	0	0	0	2	0	0	0	2

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
	business rules												
93	Scope creep	0	1	0	0	0	0	0	0	0	0	0	1
94	Simplicity of Dimensional models	0	0	1	0	0	0	0	1	0	2	0	4
95	Slowly changing dimensions awareness	0	0	0	2	0	0	0	0	0	0	0	2
96	Slowly Changing Dimensions/graceful changes	0	0	0	0	0	0	0	0	0	0	0	0
97	Standards not being used	0	1	1	1	0	1	1	1	1	1	1	9
98	Steering committee not being utilised	1	1	1	1	2	1	2	1	1	2	1	14
99	Support activity is granting of access to users	0	1	0	0	0	0	0	0	0	0	0	1
100	Team knowledge leveraged	0	0	0	1	1	0	0	0	0	0	0	2
101	Technology reliance and dependency	1	2	0	0	0	1	1	0	0	0	0	5
102	Technology used to improve performance	1	0	0	1	0	0	0	0	0	0	0	2
103	Ticket driven support process leads to communication challenge	0	0	0	0	0	0	2	0	0	0	0	2
104	Tools used for error handling	0	0	0	0	0	1	0	0	0	0	0	1
105	Training of users	0	0	0	0	0	0	0	0	0	0	0	0
106	Unaware of Inmon's partitioning strategies	2	1	1	1	1	1	1	1	0	1	0	10
107	Use of metadata	0	0	0	0	0	0	0	0	0	0	0	0
108	Use of Tools	0	0	0	0	0	0	0	0	0	0	0	0
109	User Interface challenges	0	0	0	0	0	0	0	0	2	0	0	2

		P 2: AB.docx	P 3: DN.docx	P 4: MW.docx	P 5: MV.docx	P 6: OF.docx	P 7: KS.docx	P 8: MN.docx	P 9: SM.docx	P10: AN.docx	P11: GW.docx	P12: SR.docx	TOTALS:
110	User training occurs	0	0	0	0	0	0	0	0	1	0	1	2
111	Users involved in development	0	0	0	0	0	0	0	1	0	0	0	1
112	Users need re-training as DW system not used regularly	0	0	0	0	0	0	0	0	1	2	0	3
113	Visual BI aids the relationship between business and DW team	0	1	0	0	0	0	0	0	0	0	1	2
114	Volume of data	0	0	0	0	0	0	0	0	0	0	0	0
115	Waterfall type of methodology	0	0	0	0	1	0	1	0	1	0	1	4
	Totals:	36	34	39	36	24	39	54	30	31	34	32	389

Table B - 1Code list per document

Appendix C
Links between codes

This appendix contains a list of all the codes and the links between them. An example of this is:

Code Excerpt C1 - Code neighbour list Code-Filter: All

Code: cyclical development not being practised {8-3}

<is part of> Cyclical development

<is part of> Dimensional models

<is part of> Methodology comparison

Code: Developer Training {0-6}

Challenge of learning new system <is part of>

Difficulty of using the technology <is part of>

Effectiveness of Developer training <is part of>

Learning occurs under pressure <is part of>

maintenance requires experienced professionals <is part of>

Priority on understanding business rules <is part of>

These relationships demonstrate which themes are recurring and what how they are related. The numbers between parentheses, "{}" , indicate how many times a occurs and how many links to other codes this code has. In Code Excerpt C1 above, the text "**cyclical development not being practised {8-3}**', means that the code **cyclical development not being practised** has been coded 8 times in the document set and contains 3 links to other codes. These 3 codes are:

<is part of> Cyclical development

<is part of> Dimensional models

<is part of> Methodology comparison

The complete list of codes and their relationships are listed below in Code Excerpt C2.

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: *Ad hoc* additions {2-1}

Methodology comparison <is part of>

Code: Adaptive maintenance treated as project {2-1}

Methodology comparison <is part of>

Code: Agile type of methodology {4-2}

<is part of> Cyclical development

<is part of> Methodology comparison

Code: Availability higher priority with OLTP {2-1}

<is part of> OLTP Comparison

Code: Business communication with Team {0-18}~

challenge of resource and time constraint <is part of>

Challenge of Team communication <is part of>

communication challenges of outsourcing <is part of>

dedicated user support staff <is part of>

Direct contact between business and team <is part of>

High volume require better user communication <is part of>

lack of communication between Developer and User <is part of>

lack of input from client <is part of>

Lack of integration in team <is part of>

Management does not prioritise perfective maintenance <is part of>

Outsourced developers do more harm by focusing on getting the job done quickly <is part of>

priority based on deadlines <is part of>

Priority on understanding business rules <is part of>

Scope creep <is part of>

Steering committee not being utilised <is part of>

Code Excerpt C2 - Code neighbour list Code-Filter: All

Team knowledge leveraged <is part of>

Ticket driven support process leads to communication challenge <is part of>

Visual BI aids the relationship between business and DW team <is part of>

Code: challenge of changing user mindset to use DW {2-2}

<is part of> Challenges experienced

<is part of> Training of users

Code: challenge of data integrity {2-2}

<is part of> Challenges experienced

<is part of> Preventive/Perfective maintenance

Code: challenge of duration for loading data {4-3}

<is part of> Challenges experienced

<is part of> Preventive/Perfective maintenance

<is part of> Volume of data

Code: challenge of DW methodology knowledge {1-2}

<is part of> Challenges experienced

<is part of> Dimensional models

Code: Challenge of high volume addressed through perfective maintenace {2-3}

<is part of> Challenges experienced

<is part of> Preventive/Perfective maintenance

<is part of> Volume of data

Code: Challenge of learning new system {1-2}

<is part of> Challenges experienced

<is part of> Developer Training

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: challenge of resource and time constraint {1-2}

<is part of> Business communication with Team

<is part of> Challenges experienced

Code: Challenge of Team communication {4-2}

<is part of> Business communication with Team

<is part of> Challenges experienced

Code: Challenge of updating documentation {3-2}

<is part of> Challenges experienced

<is part of> Documentation including use of metadata

Code: Challenges experienced {0-14}

challenge of changing user mindset to use DW <is part of>

challenge of data integrity <is part of>

challenge of duration for loading data <is part of>

challenge of DW methodology knowledge <is part of>

Challenge of high volume addressed through perfective maintenace <is part of>

Challenge of learning new system <is part of>

challenge of resource and time constraint <is part of>

Challenge of Team communication <is part of>

Challenge of updating documentation <is part of>

communication challenges of outsourcing <is part of>

High volumes pose challenge <is part of>

mulitple sources challenge <is part of>

Ticket driven support process leads to communication challenge <is part of>

User Interface challenges <is part of>

Code: communication challenges of outsourcing {8-3}~

<is part of> Business communication with Team

<is part of> Challenges experienced

<is part of> Outsourcing

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: Cyclical development {1-3}

Agile type of methodology <is part of>

cyclical development not being practised <is part of>

Inmon type of philosophy <is part of>

Code: cyclical development not being practised {8-3}

<is part of> Cyclical development

<is part of> Dimensional models

<is part of> Methodology comparison

Code: dedicated user support staff {1-2}

<is part of> Business communication with Team

<is part of> Methodology comparison

Code: Developer Training {0-6}

Challenge of learning new system <is part of>

Difficulty of using the technology <is part of>

Effectiveness of Developer training <is part of>

Learning occurs under pressure <is part of>

maintenance requires experienced professionals <is part of>

Priority on understanding business rules <is part of>

Code: Development tools not well suited for maintenance {1-1}

<is part of> Use of Tools

Code: Different developers doing maintenance {2-3}

<is part of> Dimensional models

<is part of> Methodology comparison

<is part of> Outsourcing

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: Difficulty of using the technology {2-4}

<is part of> Developer Training

<is part of> Documentation including use of metadata

<is part of> Experience with Preventive/Perfective maintenance

<is part of> Training of users

Code: Dimensional models {0-11}

<is part of> Slowly Changing Dimensions/graceful changes

challenge of DW methodology knowledge <is part of>

cyclical development not being practised <is part of>

Different developers doing maintenance <is part of>

Does not make use of Kimball's bus matrix when planning changes <is part of>

Does not use dimensional models <is part of>

materialized view problem <is part of>

mulitple sources challenge <is part of>

Not familiar with Dimensional models <is part of>

Simplicity of Dimensional models <is part of>

Slowly changing dimensions awareness <is part of>

Code: Direct contact between business and team {4-2}

<is part of> Business communication with Team

<is part of> Training of users

Code: Documentation availability {1-1}

<is part of> Documentation including use of metadata

Code: Documentation availability negative {5-1}

<is part of> Documentation including use of metadata

Code: Documentation availability positive {3-1}

<is part of> Documentation including use of metadata

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: Documentation given lower priority {2-1}

<is part of> Documentation including use of metadata

Code: Documentation including use of metadata {0-16}

Challenge of updating documentation <is part of>

Difficulty of using the technology <is part of>

Documentation availability <is part of>

Documentation availability negative <is part of>

Documentation availability positive <is part of>

Documentation given lower priority <is part of>

Documentation tool not useful <is part of>

documentation tool used <is part of>

Documentation tools not being used <is part of>

Documentation usefulness negative <is part of>

Documentation usefulness positive <is part of>

lengthy documentation is counter productive <is part of>

Metada used for documentation <is part of>

Metadata not used for documentation <is part of>

Metadata use is manual <is part of>

Standards not being used <is part of>

Code: Documentation tool not useful {3-1}

<is part of> Documentation including use of metadata

Code: documentation tool used {1-2}

<is part of> Documentation including use of metadata

<is part of> Use of Tools

Code: Documentation tools not being used {12-1}

<is part of> Documentation including use of metadata

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: Documentation usefulness negative {8-1}

<is part of> Documentation including use of metadata

Code: Documentation usefulness positive {4-1}

<is part of> Documentation including use of metadata

Code: Does not make use of Kimball's bus matrix when planning changes {14-2}

<is part of> Dimensional models

<is part of> Slowly Changing Dimensions/graceful changes

Code: Does not use dimensional models {4-1}

<is part of> Dimensional models

Code: DW used to ease the load on transactional system {2-2}

<is part of> Methodology comparison

<is part of> OLTP Comparison

Code: Effectiveness of Developer training {1-2}

<is part of> Developer Training

<is part of> Training of users

Code: Experience with Preventive/Perfective maintenance {0-5}

Difficulty of using the technology <is part of>

Lack of preventive/perfective maintenance <is part of>

perfective maintenance is performed <is part of>

preventive maintenance monitoring occurs <is part of>

Unaware of Inmon's partitioning strategies <is part of>

Code: Hardware improvement to cope with volume {4-2}

<is part of> Preventive/Perfective maintenance

Code Excerpt C2 - Code neighbour list Code-Filter: All

<is part of> Volume of data

Code: High cost of outsourcing {4-1}

<is part of> Outsourcing

Code: High volume require better user communication {3-2}

<is part of> Business communication with Team

<is part of> Volume of data

Code: High volumes pose challenge {15-2}

<is part of> Challenges experienced

<is part of> Volume of data

Code: High volumes require better design {8-1}

<is part of> Volume of data

Code: Inmon type of philosophy {2-3}

<is part of> Cyclical development

<is part of> Methodology comparison

<is part of> Slowly Changing Dimensions/graceful changes

Code: Kimball methodology {4-1}

<is part of> Methodology comparison

Code: lack of communication between Developer and User {13-2}

<is part of> Business communication with Team

<is part of> Training of users

Code: lack of input from client {1-1}

Code Excerpt C2 - Code neighbour list Code-Filter: All

<is part of> Business communication with Team

Code: Lack of integration in team {1-1}

<is part of> Business communication with Team

Code: Lack of preventive/perfective maintenance {8-3}

<is part of> Experience with Preventive/Perfective maintenance

<is part of> Preventive/Perfective maintenance

<is part of> Volume of data

Code: Learning occurs under pressure {1-1}

<is part of> Developer Training

Code: lengthy documentation is counter productive {3-1}

<is part of> Documentation including use of metadata

Code: Low priority of perfective maintenance {5-1}

<is part of> Preventive/Perfective maintenance

Code: maintenance requires experienced professionals {2-2}

<is part of> Developer Training

<is part of> Methodology comparison

Code: Maintenance tools not being used {9-1}

Use of Tools <is part of>

Code: Management does not prioritise perfective maintenance {1-2}

<is part of> Business communication with Team

<is part of> Preventive/Perfective maintenance

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: materialized view problem {5-1}

<is part of> Dimensional models

Code: Metada used for documentation {1-2}

<is part of> Documentation including use of metadata

<is part of> Use of metadata

Code: Metadata not understood by developer {2-1}

<is part of> Use of metadata

Code: Metadata not used by developer {6-1}

<is part of> Use of metadata

Code: Metadata not used for documentation {6-2}

<is part of> Documentation including use of metadata

<is part of> Use of metadata

Code: Metadata use is manual {2-2}

<is part of> Documentation including use of metadata

<is part of> Use of metadata

Code: Metadata used by developer {2-1}

<is part of> Use of metadata

Code: Methodology comparison {0-17}

<is part of> *Ad hoc* additions

<is part of> Adaptive maintenance treated as project

Agile type of methodology <is part of>

Code Excerpt C2 - Code neighbour list Code-Filter: All

cyclical development not being practised <is part of>

dedicated user support staff <is part of>

Different developers doing maintenance <is part of>

DW used to ease the load on transactional system <is part of>

Inmon type of philosophy <is part of>

Kimball methodology <is part of>

maintenance requires experienced professionals <is part of>

Methodology linked to technology <is part of>

priority based on deadlines <is part of>

Standards not being used <is part of>

Steering committee not being utilised <is part of>

Technology reliance and dependency <is part of>

users involved in development <is part of>

waterfall type of methodology <is part of>

Code: Methodology linked to technology {1-1}

<is part of> Methodology comparison

Code: More focus placed on perfective maintenance with OLTP {2-1}

<is part of> OLTP Comparison

Code: mulitple sources challenge {3-2}

<is part of> Challenges experienced

<is part of> Dimensional models

Code: Not answered {19-0}

Code: Not familiar with Dimensional models {1-1}

<is part of> Dimensional models

Code: not relevant {19-0}

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: OLTP Comparison {0-12}

Availability higher priority with OLTP <is part of>

DW used to ease the load on transactional system <is part of>

More focus placed on perfective maintenance with OLTP <is part of>

OLTP dev require more testing <is part of>

OLTP forces users to change <is part of>

OLTP given higher priority for maintenance <is part of>

OLTP has simpler data retrieval <is part of>

OLTP maintenance focus is on system architecture (hardware) <is part of>

OLTP maintenance focuses on release management <is part of>

OLTP maintenance less challenging as data analysis not required <is part of>

OLTP maintenance less challenging as reloads not required <is part of>

Performance more important in OLTP <is part of>

Code: OLTP dev require more testing {1-1}

<is part of> OLTP Comparison

Code: OLTP forces users to change {1-1}

<is part of> OLTP Comparison

Code: OLTP given higher priority for maintenance {1-1}

<is part of> OLTP Comparison

Code: OLTP has simpler data retrieval {2-1}

<is part of> OLTP Comparison

Code: OLTP maintenance focus is on system architecture (hardware) {1-1}

<is part of> OLTP Comparison

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: OLTP maintenance focuses on release management {1-1}

<is part of> OLTP Comparison

Code: OLTP maintenance less challenging as data analysis not required {2-1}

<is part of> OLTP Comparison

Code: OLTP maintenance less challenging as reloads not required {2-1}

<is part of> OLTP Comparison

Code: Outsourced developers do more harm by focusing on getting the job done quickly {2-2}

<is part of> Business communication with Team

<is part of> Outsourcing

Code: Outsourced developers have to spend time learning {8-1}

<is part of> Outsourcing

Code: Outsourcing {0-8}

communication challenges of outsourcing <is part of>

Different developers doing maintenance <is part of>

High cost of outsourcing <is part of>

Outsourced developers do more harm by focusing on getting the job done quickly <is part of>

Outsourced developers have to spend time learning <is part of>

Outsourcing negative <is part of>

Outsourcing pose risk of losing knowledge of a system <is part of>

Outsourcing positive <is part of>

Code: Outsourcing negative {7-1}

<is part of> Outsourcing

Code: Outsourcing pose risk of losing knowledge of a system {2-1}

Code Excerpt C2 - Code neighbour list Code-Filter: All

<is part of> Outsourcing

Code: Outsourcing positive {3-1}

<is part of> Outsourcing

Code: perfective maintenance is performed {11-3}

<is part of> Experience with Preventive/Perfective maintenance

<is part of> Preventive/Perfective maintenance

<is part of> Volume of data

Code: Performance more important in OLTP {2-1}

<is part of> OLTP Comparison

Code: preventive maintenance monitoring occurs {3-3}

<is part of> Experience with Preventive/Perfective maintenance

<is part of> Preventive/Perfective maintenance

<is part of> Volume of data

Code: Preventive/Perfective maintenance {0-11}

challenge of data integrity <is part of>

challenge of duration for loading data <is part of>

Challenge of high volume addressed through perfective maintenace <is part of>

Hardware improvement to cope with volume <is part of>

Lack of preventive/perfective maintenance <is part of>

Low priority of perfective maintenance <is part of>

Management does not prioritise perfective maintenance <is part of>

perfective maintenance is performed <is part of>

preventive maintenance monitoring occurs <is part of>

Technology used to improve performance <is part of>

Unaware of Inmon's partitioning strategies <is part of>

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: priority based on deadlines {1-2}

<is part of> Business communication with Team

<is part of> Methodology comparison

Code: Priority on understanding business rules {2-2}

<is part of> Business communication with Team

<is part of> Developer Training

Code: Scope creep {1-1}

<is part of> Business communication with Team

Code: Simplicity of Dimensional models {4-1}

<is part of> Dimensional models

Code: Slowly changing dimensions awareness {2-2}

<is part of> Dimensional models

<is part of> Slowly Changing Dimensions/graceful changes

Code: Slowly Changing Dimensions/graceful changes {0-4}

Dimensional models <is part of>

Does not make use of Kimball's bus matrix when planning changes <is part of>

Inmon type of philosophy <is part of>

Slowly changing dimensions awareness <is part of>

Code: Standards not being used {9-2}

<is part of> Documentation including use of metadata

<is part of> Methodology comparison

Code: Steering committee not being utilised {14-2}

<is part of> Business communication with Team

Code Excerpt C2 - Code neighbour list Code-Filter: All

<is part of> Methodology comparison

Code: Support activity is granting of access to users {1-1}

<is part of> Training of users

Code: Team knowledge leveraged {2-2}

<is part of> Business communication with Team

<is part of> Training of users

Code: Technology reliance and dependency {5-1}

<is part of> Methodology comparison

Code: Technology used to improve performance {2-2}

<is part of> Preventive/Perfective maintenance

<is part of> Use of Tools

Code: Ticket driven support process leads to communication challenge {2-2}

<is part of> Business communication with Team

<is part of> Challenges experienced

Code: Tools used for error handling {1-1}

<is part of> Use of Tools

Code: Training of users {0-10}

challenge of changing user mindset to use DW <is part of>

Difficulty of using the technology <is part of>

Direct contact between business and team <is part of>

Effectiveness of Developer training <is part of>

lack of communication between Developer and User <is part of>

Support activity is granting of access to users <is part of>

Code Excerpt C2 - Code neighbour list Code-Filter: All

Team knowledge leveraged <is part of>

User Interface challenges <is part of>

User training occurs <is part of>

Users need re-training as DW system not used regularly <is part of>

Code: Unaware of Inmon's partitioning strategies {10-3}

<is part of> Experience with Preventive/Perfective maintenance

<is part of> Preventive/Perfective maintenance

<is part of> Volume of data

Code: Use of metadata {0-6}

Metada used for documentation <is part of>

Metadata not understood by developer <is part of>

Metadata not used by developer <is part of>

Metadata not used for documentation <is part of>

Metadata use is manual <is part of>

Metadata used by developer <is part of>

Code: Use of Tools {0-6}

<is part of> Maintenance tools not being used

Development tools not well suited for maintenance <is part of>

documentation tool used <is part of>

Technology used to improve performance <is part of>

Tools used for error handling <is part of>

Visual BI aids the relationship between business and DW team <is part of>

Code: User Interface challenges {2-2}

<is part of> Challenges experienced

<is part of> Training of users

Code: User training occurs {2-1}

<is part of> Training of users

Code Excerpt C2 - Code neighbour list Code-Filter: All

Code: users involved in development {1-1}

<is part of> Methodology comparison

Code: Users need re-training as DW system not used regularly {3-1}

<is part of> Training of users

Code: Visual BI aids the relationship between business and DW team {2-2}

<is part of> Business communication with Team

<is part of> Use of Tools

Code: Volume of data {0-10}

challenge of duration for loading data <is part of>

Challenge of high volume addressed through perfective maintenace <is part of>

Hardware improvement to cope with volume <is part of>

High volume require better user communication <is part of>

High volumes pose challenge <is part of>

High volumes require better design <is part of>

Lack of preventive/perfective maintenance <is part of>

perfective maintenance is performed <is part of>

preventive maintenance monitoring occurs <is part of>

Unaware of Inmon's partitioning strategies <is part of>

Code: waterfall type of methodology {4-1}

<is part of> Methodology comparison

Appendix D
Code Families

Code Excerpt D1 contains all code families and the codes that belong to them

Code Excerpt D1 - Code Families

Code Families

HU: Shakeel_individual files
File: [C:\Users\Shakeel\Google Drive\Atlas TI\Shakeel_individual files.hpr7]
Edited by: Super
Date/Time: 2014-05-19 02:14:27

Code Family: Challenges experienced
Created: 2014-02-21 03:50:55 (Super)
Codes (15): [challenge of changing user mindset to use DW] [challenge of data integrity] [challenge of duration for loading data] [challenge of DW methodology knowledge] [Challenge of high volume addressed through perfective maintenance] [Challenge of learning new system] [challenge of resource and time constraint] [Challenge of Team communication] [Challenge of updating documentation] [Challenges experienced] [communication challenges of outsourcing] [High volumes pose challenge] [mulitple sources challenge] [Ticket driven support process leads to communication challenge] [User Interface challenges]
Quotation(s): 44

Code Family: Communication
Created: 2013-11-21 23:11:23 (Super)
Codes (19): [Business communication with Team] [challenge of resource and time constraint] [Challenge of Team communication] [communication challenges of outsourcing] [dedicated user support staff] [Direct contact between business and team] [High volume require better user communication] [lack of communication between Developer and User] [lack of input from client] [Lack of integration in team] [Management does not prioritise perfective maintenance] [Outsourced developers do more harm by focusing on getting the job done quickly] [priority based on deadlines] [Priority on understanding business rules] [Scope creep] [Steering committee not being utilised] [Team knowledge leveraged] [Ticket driven support process leads to communication challenge] [Visual BI aids the relationship between business and DW team]
Quotation(s): 55

Code Family: Cyclical development
Created: 2013-11-21 23:38:39 (Super)
Codes (4): [Agile type of methodology] [Cyclical development] [cyclical development not being practised] [Inmon type of philosophy]
Quotation(s): 14

Code Family: Dimensional modelling
Created: 2013-11-24 22:00:24 (Super)
Codes (14): [*Ad hoc* additions] [challenge of DW methodology knowledge] [cyclical development not being practised] [Different developers doing maintenance] [Dimensional models] [Does not make use of Kimball's bus matrix when planning changes] [Does not use dimensional models] [Kimball methodology] [materialized view problem] [mulitple sources challenge] [Not familiar with Dimensional models] [Simplicity of Dimensional models] [Slowly changing dimensions awareness] [Slowly

Changing Dimensions/graceful changes]
Quotation(s): 43

Code Family: Documentation
Created: 2013-11-25 21:15:10 (Super)
Codes (17): [Challenge of updating documentation] [Difficulty of using the technology] [Documentation availability] [Documentation availability negative] [Documentation availability positive] [Documentation given lower priority] [Documentation including use of metadata] [Documentation tool not useful] [documentation tool used] [Documentation tools not being used] [Documentation usefulness negative] [Documentation usefulness positive] [lengthy documentation is counter productive] [Metada used for documentation] [Metadata not used for documentation] [Metadata use is manual] [Standards not being used]
Quotation(s): 46

Code Family: High Volumes
Created: 2013-11-25 21:10:53 (Super)
Codes (11): [challenge of duration for loading data] [Challenge of high volume addressed through perfective maintenance] [Hardware improvement to cope with volume] [High volume require better user communication] [High volumes pose challenge] [High volumes require better design] [Lack of preventive/perfective maintenance] [perfective maintenance is performed] [preventive maintenance monitoring occurs] [Unaware of Inmon's partitioning strategies] [Volume of data]
Quotation(s): 51

Code Family: Metadata
Created: 2013-11-25 22:17:45 (Super)
Codes (7): [Metada used for documentation] [Metadata not understood by developer] [Metadata not used by developer] [Metadata not used for documentation] [Metadata use is manual] [Metadata used by developer] [Use of metadata]
Quotation(s): 17

Code Family: Methodology
Created: 2013-11-24 22:13:45 (Super)
Codes (19): [*Ad hoc* additions] [Adaptive maintenance treated as project] [Agile type of methodology] [cyclical development not being practised] [dedicated user support staff] [Different developers doing maintenance] [DW used to ease the load on transactional system] [Inmon type of philosophy] [Kimball methodology] [Lack of integration in team] [maintenance requires experienced professionals] [Methodology comparison] [Methodology linked to technology] [priority based on deadlines] [Standards not being used] [Steering committee not being utilised] [Technology reliance and dependency] [users involved in development] [waterfall type of methodology]
Quotation(s): 52

Code Family: Not applicable
Created: 2013-12-19 21:46:06 (Super)
Codes (2): [Not answered] [not relevant]
Quotation(s): 38

Code Family: OLTP Comparison
Created: 2014-02-21 03:48:01 (Super)
Codes (13): [Availability higher priority with OLTP] [DW used to ease the load on transactional system] [More focus placed on perfective maintenance with OLTP] [OLTP Comparison] [OLTP dev require more testing] [OLTP forces users to change] [OLTP given higher priority for maintenance] [OLTP has simpler data retrieval] [OLTP maintenance focus is on system architecture (hardware)] [OLTP maintenance focuses on release management] [OLTP maintenance less challenging as data analysis not required] [OLTP maintenance less challenging as reloads not required] [Performance more important in OLTP]
Quotation(s): 11

Code Family: Outsourcing
Created: 2013-11-21 23:32:06 (Super)
Codes (9): [communication challenges of outsourcing] [Different developers doing maintenance] [High cost of outsourcing] [Outsourced developers do more harm by focusing on getting the job done quickly] [Outsourced developers have to spend time learning] [Outsourcing] [Outsourcing negative] [Outsourcing pose risk of losing knowledge of a system] [Outsourcing positive]
Quotation(s): 21

Code Family: parent codes
Created: 2014-03-02 19:45:33 (Super)
Codes (15): [Business communication with Team] [Challenges experienced] [Developer Training] [Dimensional models] [Documentation including use of metadata] [Experience with Preventive/Perfective maintenance] [Methodology comparison] [OLTP Comparison] [Outsourcing] [Preventive/Perfective maintenance] [Slowly Changing Dimensions/graceful changes] [Training of users] [Use of metadata] [Use of Tools] [Volume of data]
Quotation(s): 0

Code Family: Preventive/Perfective maintenance
Created: 2013-11-25 21:09:01 (Super)
Codes (14): [challenge of data integrity] [challenge of duration for loading data] [Challenge of high volume addressed through perfective maintenance] [Difficulty of using the technology] [Experience with Preventive/Perfective maintenance] [Hardware improvement to cope with volume] [Lack of preventive/perfective maintenance] [Low priority of perfective maintenance] [Management does not prioritise perfective maintenance] [perfective maintenance is performed] [preventive maintenance monitoring occurs] [Preventive/Perfective maintenance] [Technology used to improve performance] [Unaware of Inmon's partitioning strategies]
Quotation(s): 48

Code Family: Slowly changing dimensions
Created: 2013-11-25 22:31:59 (Super)
Codes (4): [Does not make use of Kimball's bus matrix when planning changes] [Inmon type of philosophy] [Slowly changing dimensions awareness] [Slowly Changing Dimensions/graceful changes]
Quotation(s): 17

Code Family: Training
Created: 2013-11-24 22:24:54 (Super)
Codes (16): [challenge of changing user mindset to use DW] [Challenge of learning new system] [Developer Training] [Difficulty of using the technology] [Direct contact between business and team] [Effectiveness of Developer training] [lack of communication between Developer and User] [Learning occurs under pressure] [maintenance requires experienced professionals] [Priority on understanding business rules] [Support activity is granting of access to users] [Team knowledge leveraged] [Training of users] [User Interface challenges] [User training occurs] [Users need re-training as DW system not used regularly]
Quotation(s): 33

Code Family: Use of Tools
Created: 2013-11-25 22:30:38 (Super)
Codes (8): [Development tools not well suited for maintenance] [documentation tool used] [Documentation tools not being used] [Maintenance tools not being used] [Technology used to improve performance] [Tools used for error handling] [Use of Tools] [Visual BI aids the relationship between business and DW team]
Quotation(s): 27