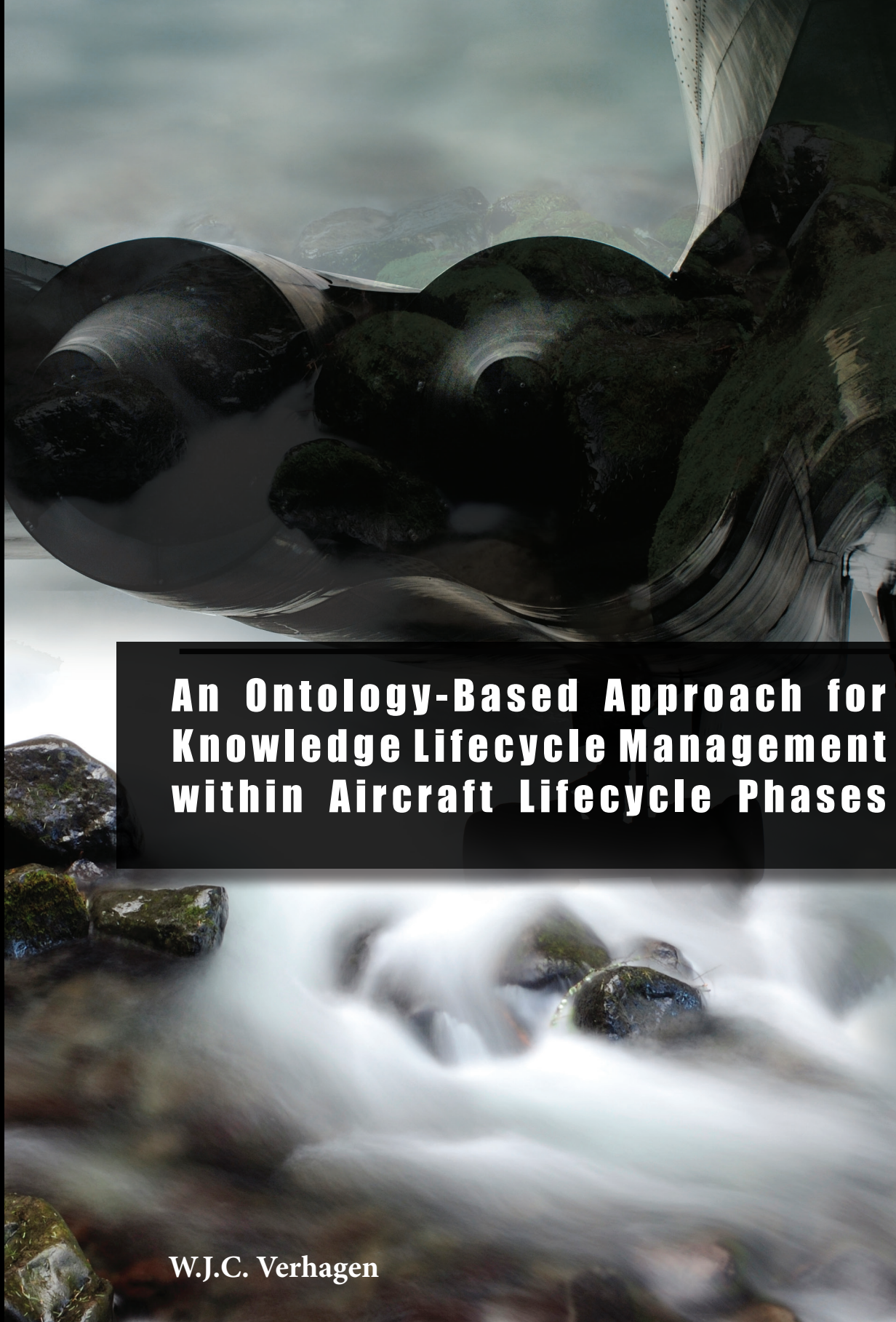

An Ontology-Based Approach for Knowledge Lifecycle Management within Aircraft Lifecycle Phases

W.J.C. Verhagen

An Ontology-Based Approach for Knowledge Lifecycle Management within Aircraft Lifecycle Phases W.J.C. Verhagen

An Ontology-Based Approach for Knowledge Lifecycle Management within Aircraft Lifecycle Phases

An Ontology-Based Approach for Knowledge Lifecycle Management within Aircraft Lifecycle Phases

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op donderdag 18 juli 2013 om 10:00 uur

door

Wilhelmus Johannes Cornelis VERHAGEN
ingenieur in de luchtvaart en ruimtevaart
geboren te Moergestel

Dit proefschrift is goedgekeurd door de promotor:

Prof.dr. R. Curran

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr. R. Curran,	Technische Universiteit Delft, promotor
Prof.dr. J. Scanlan	University of Southampton, United Kingdom
Prof.dr. T. Tomiyama	Cranfield University, United Kingdom
Prof.dr. G. Lodewijks	Technische Universiteit Delft
Dr. J-P. Clarke	Georgia Institute of Technology
Dr. G. La Rocca	Technische Universiteit Delft
Dr. P. Bermell-Garcia	EADS Innovation Works
Prof. dr. D.G. Simons	Technische Universiteit Delft, reservelid

ISBN 978-90-8891-659-5

Keywords: Knowledge Lifecycle, Knowledge Based Engineering, Knowledge Based Applications, Ontology

Published by Uitgeverij BOXPress, 's-Hertogenbosch.

Dedicated to my mother Christine, in loving memory.

Table of Contents

Acknowledgements ..... **xi**

Summary ..... **xiii**

List of Figures ..... **xvii**

List of Tables ..... **xxi**

Nomenclature ..... **xxiii**

1 Introduction **1**
1.1 Challenges in Knowledge Engineering for the Aircraft Lifecycle 1
1.2 Research Approach 2
1.2.1 Research Framework 3
1.2.2 Research Design 6
1.3 Dissertation Structure 8

2 Exploration of the Research Context **11**
2.1 Perspectives on Knowledge and Knowledge Change 11
2.2 State of the Art and Challenges for Knowledge Perspectives along the Product Lifecycle 20
2.2.1 Data through Product Life: Product Data Management 23
2.2.2 Information through Product Life: Product Lifecycle Management 25
2.2.3 Knowledge through Product Life: Knowledge Management & Knowledge Engineering 29
2.3 Identification of Research Challenges 36

3 Theory Development **41**
3.1 A Conceptual Model for the Lifecycle of Knowledge 42
3.1.1 State of the Art and Shortcomings of Knowledge Lifecycle research 42
3.1.2 Requirements on Definition of a Knowledge Lifecycle Model 45
3.1.3 Research Contribution 1: Conceptual Knowledge Lifecycle Model 47
3.1.4 Concluding Remarks 49
3.2 A Model-Based Approach to Support Knowledge Change: the Knowledge Lifecycle Ontology 51
3.2.1 The Role of Ontologies in Supporting Knowledge-Based Applications through Knowledge Life: State of the Art and Shortcomings 51
3.2.2 Main Elements for the Development of the Knowledge Life Cycle Ontology 58
3.2.3 Research Contribution 2: Knowledge Life Cycle Ontology 70
3.2.4 Concluding Remarks 77
3.3 The KNOMAD Methodology for Supporting KBS Development incorporating Knowledge Change 80

3.3.1 State of the Art in Methodologies for KBS development 80
3.3.2 Shortcomings of Existing Methodologies and Associated Research Requirements 85
3.3.3 Research Contribution 3: KNOMAD Methodology 86
3.3.4 Concluding Remarks 90
3.4 Discussion of Contributions 91
3.4.1 Discussion of the Knowledge Lifecycle Model 91
3.4.2 Discussion of the Knowledge Lifecycle Ontology 92
3.4.3 Discussion of the KNOMAD methodology 93
3.5 Proposing a Case Study approach 94

4 Design Case Study: Ply Stacking Sequence Optimization for Composite Wing Panels 99
4.1 Case Study Context and Challenges 99
4.2 Application of Theory to Design Case Study 104
4.2.1 Application of Knowledge Lifecycle Model: Identifying Knowledge Change 105
4.2.2 Application of KLC Ontology: Task Analysis 107
4.2.3 Application of KNOMAD: Solution Approach 109
4.3 Results 110
4.3.1 Knowledge Capture & Identification of Knowledge Change 110
4.3.2 Normalization 112
4.3.3 Organisation 113
4.3.4 Modelling & Implementation 117
4.3.5 Analysis & Delivery 127
4.4 Discussion of Results 128

5 Manufacturing Case Study: Composite Wing Cost Modelling & Estimation 129
5.1 Case Study Context and Challenges 130
5.2 Application of Theory to Manufacturing Case Study 136
5.2.1 Application of Knowledge Lifecycle Model: Identifying Knowledge Change 136
5.2.2 Application of KLC Ontology: Task Analysis 138
5.2.3 Application of KNOMAD: Solution Approach 140
5.3 Results 141
5.3.1 Knowledge Identification & Capture 142
5.3.2 Normalization 144
5.3.3 Organisation 146
5.3.4 Modelling & Implementation 150
5.3.5 Analysis & Delivery 158
5.4 Discussion of Results 159

6 Maintenance Case Study: Supporting Wing Maintenance – B737 Leading Edge Slat Downstop Assembly Modification & Inspection 161
6.1 Case Study Context and Challenges 161

6.2 Application of Theory to Maintenance Case Study 165
6.2.1 Application of Knowledge Lifecycle Model: Identifying Knowledge Change 166
6.2.2 Application of Knowledge Lifecycle Model: Quantifying Knowledge Change 168
6.2.3 Application of KLC Ontology: Task Analysis 182
6.2.4 Application of KNOMAD: Solution Approach 183
6.3 Results 184
6.3.1 Knowledge Identification & Capture 185
6.3.2 Normalization 188
6.3.3 Organisation 188
6.3.4 Modelling & Implementation 191
6.3.5 Analysis & Delivery 202
6.4 Discussion of Results 202

7 Conclusion 205
7.1 Research Synthesis 205
7.1.1 Synthesizing a Vision for Knowledge Engineering 205
7.1.2 Synthesizing the Case Study Results relative to Research Objectives and Challenges 208
7.2 Research Conclusions 210
7.2.1 Theory Development: Knowledge Lifecycle Modelling 211
7.2.2 Theory Development: Ontology-based Approach to Support Knowledge Change 212
7.2.3 Theory Development: Methodology Development 213
7.3 Research Limitations & Recommendations 214

References 219

Appendix A: Complexity Estimation 229

Samenvatting 233

Curriculum Vitae 237

List of Publications 238

Acknowledgements

Obtaining a PhD is often seen as an individual achievement. While in the end the responsibility falls upon the shoulders of the PhD candidate, the journey towards the PhD is most emphatically a shared one. It is the people you meet and interact with during the PhD journey that make it such a memorable and worthwhile experience. I wish to thank several people who have made the journey so much easier to complete.

First, I wish to express my deep gratitude to my promotor Ricky Curran for his guidance and unwavering support, both research-related and personal. Ricky, I very much enjoyed working together on research and deeply appreciate the opportunities you have given me to broaden my horizon through educational and international activities. I am looking forward with great anticipation towards building upon our joint ambitions for the Air Transport & Operations chair.

I also wish to extend my gratitude to my colleagues from industry. Christian, thank you for that vital first push. Working together with the Ardans colleagues (Jean-Pierre, Alain and above all Pierre) has been a pleasure. I very much appreciate the hospitality, expertise and support extended by the members of the EADS IW team involved in parts of the research: Domingo, Simon, Kiran, Jean-Luc, Gary, Alistair, Romaric and Jean-Baptiste. Pablo must be singled out in his vital role as industrial advisor-of-sorts during the critical stage of the research process. Pablo, your professional and personal character is of the highest quality. It was an honour and pleasure to work with you and the EADS team.

Thanks to the members of the ATO staff for making the daily work environment so pleasant. The regular and irregular members of the 'lunch club' make for stimulating lunch time conversation, ranging from aerospace to distinctly non-aerospace related topics. A warm thanks to Liza, Geeta and Vera for being the unsung yet vital heroines of the group.

I am deeply indebted to my friends from Jochvipelisawi and my family for their unconditional support. Special thanks to Marieke for her fantastic cover design! It is however three persons that deserve a very special mention. Lisette, you are the love of my life – I couldn't have done the PhD without you. Finally, the support of my father Jan and mother Christine is and has been the best source of motivation during these and previous years.

Summary

An Ontology-Based Approach for Knowledge Lifecycle Management within Aircraft Lifecycle Phases

In the aerospace domain, manufacturers and operators constantly seek to improve their products and processes. Increasingly, knowledge-based applications are developed to support or automate knowledge-intensive engineering tasks, saving time and money. However, engineering knowledge is likely to change over time, which has implications for knowledge-based applications.

A central challenge to consider is related to the nature of knowledge and its behaviour over time. Does knowledge change and therefore, does it have a lifecycle of its own? With respect to the issue of *knowledge change*, current research is rather limited. Various authors (e.g. Schorlemmer *et al.* (2002), Alavi and Leidner (2001), Stokes (2001), Nonaka *et al.* (2000) and Schreiber *et al.* (1999)) indicate that knowledge changes, but these authors do not accurately define their concepts, most do not back up their assertions, and none go beyond a qualitative assessment of knowledge change.

This has major ramifications from a practical perspective. If knowledge changes, existing knowledge-based applications risk becoming rapidly obsolete. Coenen and Bench-Capon (1993) offer an indication of the magnitude of the problem of knowledge change: the knowledge-based system that was studied incorporated an estimated 50% change in rules on a yearly basis, while the overall knowledge base expanded about fourfold in the first 3 years of operation. Van Dijk *et al.* (2012) offers an indication of the costs associated with maintaining a knowledge-based application to keep functionality and knowledge up to date, which are estimated to be 25% of non-recurring software development cost *on a yearly basis*.

How can knowledge-based applications cope with knowledge change? It is necessary to develop models and methods to enable the development of more robust engineering applications: *usability* and *maintainability* of knowledge and knowledge-based applications must be facilitated. The following high-level research goal is consequently identified:

> *Support consistent formalization, use and maintenance of changing knowledge within aircraft lifecycle phases to improve domain-specific modelling, execution and control of engineering tasks*

Knowledge change is defined here as a change in knowledge over time, where knowledge is defined as processed information with a capability for effective

action. Consequently, the following types of change may be discerned in a knowledge-based application: changes in values (**data change**), changes in the structured context of a knowledge element (**information change**) and changes to the capability for effective action associated with a knowledge element (**knowledge change**), where the latter can be caused by changes in rules, logic structures or attribute sets.

To achieve the high-level research goal, several contributions to theory have been developed which involved addressing associated research challenges, as shown in Table S.1.

Table S.1: Contributions to theory related to research challenges

Research contribution	Associated research challenge(s)
Knowledge Lifecycle Model	Characterise, model and quantify the behaviour of knowledge within product life
Ontology-based approach to support knowledge change: Knowledge Lifecycle Ontology	Maintainability: - Moving beyond black-box KBS applications and ensuring transparency Usability: - Task orientation - Expert / end user involvement
Methodology development: KNOMAD methodology	Methodological approach to facilitate knowledge change management

The Knowledge Lifecycle model has been developed to characterize and model the lifecycle of knowledge elements by incorporating the concepts of knowledge states and actions. In particular, the actions – including *create, formalize, use, maintain, update* and *retire* – offer the ability to meaningfully quantify knowledge behaviour over time. Through offering this capability, the Knowledge Lifecycle model goes beyond state-of-the-art in theory.

The developed Knowledge Lifecycle (KLC) ontology can serve in a structure-preserving approach towards the development, use and maintenance of knowledge-based applications. The KLC ontology revolves around two central perspectives: the Enterprise Knowledge Resource (EKR) concept in combination with an annotation structure based on the Product-Process-Resource (PPR) paradigm. An EKR is a task-oriented container representation encompassing knowledge elements, process elements and task output in the form of case reports. In combination with the PPR paradigm, 'white-box' knowledge-based applications with increased transparency can be developed. The KLC ontology moves beyond state-of-the-art through four ways: enabling structure-preserving modelling *and implementation*, representing knowledge related to individual task level, offering consistent annotation through PPR classes related to individual tasks and offering systematic storage of task outputs.

The third and final contribution to theory is the KNOMAD methodology. KNOMAD has been introduced as a methodology for the development of knowledge-based applications that can cope with changing knowledge. This methodology consists of six steps: Knowledge Capture & Identification of Knowledge Change, Normalisation, Organisation, Modelling & Implementation, Analysis and Delivery. The critical aspect of knowledge change (and associated maintenance) is accounted for by the characterisation and analysis of knowledge change at the start of the KNOMAD process. Furthermore, the organisation step emphasizes modelling of the domain knowledge layer which can subsequently be used in the Modelling & Implementation step for annotation of engineering tasks. This step also advises the use of the KLC ontology. As such, domain and task ontologies are developed and implemented as the backbone of the developed knowledge-based solutions. Consequently, the KNOMAD steps realize an ontology-based approach that addresses the research challenges of moving beyond black-box applications and ensuring transparency, task orientation and end user/expert involvement. It goes beyond existing theory by offering explicit support for knowledge change, by incorporating usability and maintainability considerations and through explicit support for assessment of knowledge-based application performance.

For validation, the ontology-based approach and the associated models and methodology have been applied in three case studies considering engineering tasks for specific aircraft life cycle phases – design, manufacturing and maintenance.

The Knowledge Lifecycle model has been successfully applied to characterise knowledge change in the design and manufacturing domains. Furthermore, the model has been applied in the maintenance domain to quantify knowledge change.

The KLC ontology has been applied in all case studies. The associated *maintainability* challenge – *moving beyond black-box and ensuring transparency* – has been addressed through the KLC ontology concepts. Through the EKR concept, traceability is ensured. In particular, the **Case** class and the associated case reports enable tracing the outputs of knowledge application for a specific task, as well as tracing the knowledge and processes used to perform a task. The metadata that is associated with knowledge and process elements (authorship, lifecycle state, status, etc.) also aids traceability in terms of knowledge ownership, validity and reliability. Through the PPR paradigm, visibility of key concepts is ensured. It has been shown in the three case studies how development of a domain-specific extension of the PPR classes facilitates semantic annotation of implemented EKRs, making it easy to find, inspect and use knowledge-based applications and their components. The *usability* challenges – *task orientation* and *expert/end user involvement* – are met through the EKR concept and the use of a

web-based knowledge management solution. In each case study, one or more EKRs have been developed to represent and support the execution of specific engineering tasks. The chosen web-based architecture facilitates user interaction with EKRs and their constituent elements (knowledge elements, process elements, cases).

The KNOMAD methodology has been applied in all three case studies. All steps of the methodology have been successfully applied to develop and implement knowledge-based applications that can handle knowledge change.

The contributions of this dissertation – Knowledge Lifecycle model, KLC ontology, KNOMAD methodology – can be expanded and refined in various ways. Most notably, the Knowledge Lifecycle model has to be quantitatively validated across more domains. Ideally, it would also be given a formal mathematical foundation. Furthermore, modelling of task complexity and hierarchies has not been performed as part of this dissertation. Finally, adding formal expressions to the KLC ontology would facilitate the use of reasoning capabilities in the development and maintenance of knowledge-based applications.

List of Figures

Figure 1.1: Research framework 7
Figure 1.2: Research roadmap 9
Figure 2.1: Data, information and knowledge transformation processes (adapted from Hicks *et al.* (2002)) 15
Figure 2.2: Selection of a material – baseline state 17
Figure 2.3: Selection of a material – changed state 17
Figure 2.4: Product lifecycle stages (adapted from Jun *et al.* (2007)) 22
Figure 2.5: Research domains of interest 23
Figure 2.6: Information flows through product life (Jun *et al.*, 2007) 27
Figure 2.7: Use of PDM and PLM throughout the product lifecycle (Lee *et al.*, 2008) 28
Figure 2.8: Positioning of knowledge disciplines (La Rocca, 2012) 30
Figure 2.9: Selection, classification and review process 34
Figure 3.1: Knowledge Lifecycle Model with knowledge states and actions 49
Figure 3.2: PROMISE Semantic Object Model (Tomasella *et al.*, 2006) 60
Figure 3.3: PDW Core Ontology (Brandt *et al.*, 2008) 61
Figure 3.4: Generic IDEF0 diagram (National Institute of Standards and Technology, 1993) 64
Figure 3.5: UML class diagram of Enterprise Knowledge Resource (Bermell-Garcia *et al.*, 2012) 65
Figure 3.6: High-level concepts and relationships of the KLC ontology 71
Figure 3.7: UML class diagram of KLC ontology 74
Figure 3.8: UML class diagram of Enterprise Knowledge Resource as implemented in the KLC ontology 77
Figure 3.9: CommonKADS methodology overview 81
Figure 3.10: KBE System Lifecycle (adapted from Oldham *et al.* (1998)) 83
Figure 3.11: MOKA methodology elements 84
Figure 3.12: KNOMAD methodology overview 87
Figure 4.1: Cross-sectional view of ply stacking sequences for two adjacent grid cells 101
Figure 4.2: Interleaved plies across cell boundary 102
Figure 4.3: Adding manufacturing considerations into a structural view of the design 102
Figure 4.4: Grid representation of a wing cover skin panel 103
Figure 4.5: Possible production issues arising from minimum course length (Blom, 2010) 106
Figure 4.6: IDEF0 A-0 diagram for composite wing cover optimization task 108
Figure 4.7: IDEF0 A0 diagram for optimization subtasks 109
Figure 4.8: Application of KNOMAD to design case study – flow chart 110
Figure 4.9: Example of manufacturing constraint stored in AKM 113
Figure 4.10: Domain-specific hierarchy for Product class 114
Figure 4.11: Domain-specific hierarchy for Process class 115
Figure 4.12: Domain-specific hierarchy for Resource class 116
Figure 4.13: EKR class diagram (UML) for design case study 119

Figure 4.14: Knowledge framework containing the eLBD and xLBD elements (Bermell-Garcia *et al.*, 2012) 121
Figure 4.15: AKM model for Enterprise_Knowledge_Resource class 122
Figure 4.16: Partial overview of implemented design and manufacturing constraints (Bermell-Garcia *et al.*, 2012) 123
Figure 4.17: Case reports (left) and their relation with design inputs and underlying knowledge (Bermell-Garcia *et al.*, 2012) 125
Figure 4.18: Annotation of the ply continuity optimization EKR 126
Figure 5.1: Standard cost modelling approach 133
Figure 5.2: Cost model evolution 137
Figure 5.3: IDEF0 A-0 diagram for cost modelling and estimation task 139
Figure 5.4: IDEF0 A0 diagram for cost modelling and estimation subtasks 140
Figure 5.5: Application of KNOMAD to manufacturing case study – flow chart 141
Figure 5.6: Example of imported geometry data for a composite wing top cover 143
Figure 5.7: Example of process parameters 144
Figure 5.8: Cost estimation output format 144
Figure 5.9: Example of cost model element stored in AKM 145
Figure 5.10: extended Product class hierarchy for the manufacturing domain 147
Figure 5.11: extended Process class hierarchy for the manufacturing domain 149
Figure 5.12: extended Resource class hierarchy for the manufacturing domain 150
Figure 5.13: Managed cost model evolution 151
Figure 5.14: Process model for CFRP T-stringer production EKR 152
Figure 5.15: EKR class diagram (UML) for manufacturing case study 153
Figure 5.16: Semantic annotation of a cost model EKR 154
Figure 5.17: Implementation architecture 155
Figure 5.18: User process for cost model composition using proof-of-concept solution . 157
Figure 5.19: Annotation of the CFRP T-stringer EKR 157
Figure 6.1: Adoption of PLM in the MRO domain (adapted from Lee *et al.* (2008)) 162
Figure 6.2: The aircraft MRO environment (adapted from Lampe *et al.* (2004)) 163
Figure 6.3: Frequency of knowledge actions (A320) 172
Figure 6.4: Knowledge actions per year versus the A320 lifetime (years) 173
Figure 6.5: Knowledge change versus lifetime (A320) 174
Figure 6.6: Frequency of knowledge actions (B737) 175
Figure 6.7: Knowledge actions per year versus the B737 lifetime (years) 176
Figure 6.8: Knowledge change versus the B737 lifetime (years) 177
Figure 6.9: Through-life implications of knowledge change (A320 sample) 180
Figure 6.10: Through-life implications of knowledge change (B737 sample) 180
Figure 6.11: IDEF0 A-0 diagram for B737 slat track main downstop modification and inspection task 182
Figure 6.12: IDEF0 A0 diagram for B737 slat track main downstop subtasks 183
Figure 6.13: Application of KNOMAD to maintenance case study – flow chart 184
Figure 6.14: Slat main track downstop assembly (FAA, 2011) 185
Figure 6.15: Aft side guide bolts (Boeing, 2010) 186
Figure 6.16: Maintenance process for modification and inspection (based on Boeing (2010)) 187

Figure 6.17: extended Product class hierarchy for the maintenance domain 189
Figure 6.18: extended Process class hierarchy for the maintenance domain 190
Figure 6.19: extended Resource class hierarchy for the maintenance domain 191
Figure 6.20: EKR class diagram (UML) for maintenance case study 193
Figure 6.21: AKM model for the Knowledge_Element class for maintenance case study 194
Figure 6.22: Example of EKR article for maintenance case study 196
Figure 6.23: Example of knowledge element article 197
Figure 6.24: Example of process element article 198
Figure 6.25: Example of case report article 199
Figure 6.26: Semantic annotation of EKR 200
Figure 6.27: Tagging an EKR in Ardans Knowledge Maker 201
Figure 7.1: Two streams of knowledge engineering related to knowledge change 207
Figure A.1: Element interactions in constrained (left) and unconstrained (right) form 229
Figure A.2: Plot of element interactions for functions *g* and *h*, where *n*=1..10, *m*=1..10 . 230
Figure A.3: Number of element interactions for *g* and *h*, where *n* = 1..10, *m* = 1..10 230

List of Tables

Table S.1: Contributions to theory related to research challenges xiv
Table 2.1: Examples of data, information and knowledge change and implications 19
Table 2.2: Summary of selected KBE development efforts 33
Table 2.3: Research objectives related to research challenges 39
Table 3.1: Research challenges related to research contributions 41
Table 3.2: Process-oriented (organisational) knowledge lifecycle models (adapted from Maksimovic *et al.* (2011)) 43
Table 3.3: Potential knowledge states 45
Table 3.4: Challenges and associated requirements on the model-based approach 52
Table 3.5: Ontology requirements 57
Table 3.6: KLC ontology requirements in relation with building blocks 67
Table 3.7: Relationships between main concepts of KLC ontology 76
Table 3-8: KLC ontology requirements versus functionality 78
Table 4.1: Captured design and manufacturing constraints 112
Table 4.2: Relationships in the design domain ontology 117
Table 5.1: Product cost modelling and estimation techniques (adapted from Niazi *et al.* (2006)) 130
Table 5.2: Assessment matrix for traditional cost estimation methods (Curran *et al.*, 2004) 131
Table 5.3: Disciplines versus fidelity (adapted from Price *et al.* (2006)) 132
Table 5.4: Example cost modelling approach: T-stringer Production 142
Table 6.1: Bivariate correlation - knowledge actions per year versus lifetime (A320) 173
Table 6.2: Bivariate correlation - knowledge change versus lifetime (A320) 174
Table 6.3: Bivariate correlation - knowledge actions per year versus lifetime (B737) 176
Table 6.4: Bivariate correlation for knowledge change versus lifetime (B737) 177
Table 6.5: Correlation results for knowledge action 'create' versus lifetime (B737) 178
Table 6.6: Correlation results for knowledge action 'maintain' versus lifetime (B737) 178
Table 6.7: Correlation results for knowledge action 'update' versus lifetime (B737) 178
Table 7.1: Research objectives related to research challenges 208
Tabel S.1: Bijdrages aan theorie en geassocieerde onderzoeksuitdagingen 234

Nomenclature

AI	Artificial Intelligence
BOL	Beginning Of Life
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CommonKADS	Common Knowledge Acquisition and Documentation Structuring
CPACS	Common Parametric Aircraft Configuration Schema
DFM	Design for Manufacturing
EBOM	Engineering Bill of Materials
EDM	Engineering Data Management
EKR	Enterprise Knowledge Resource
EOL	End Of Life
GTO	General Technology Ontology
ICARE	Illustrations, Constraints, Activities, Rules, Entities
KBE	Knowledge Based Engineering
KBS	Knowledge Based Systems
KE	Knowledge Engineering
KIF	Knowledge Interchange Format
KLC	Knowledge Lifecycle Ontology
KNOMAD	(K)nowledge Capture & Identification of Knowledge Change, (N)ormalisation, (O)rganisation, (M)odelling & Implementation, (A)nalysis and (D)elivery.
KM	Knowledge Management
MANDATE	MANufacturing DATa Exchange
MBOM	Manufacturing Bill Of Materials
MIKE	Model-based and Incremental Knowledge Engineering
MOKA	Methodology and software tools Oriented to Knowledge-based engineering Applications
MOL	Middle Of Life
OCL	Object Constraint Language
OWL	Web Ontology Language
PBS	Product Breakdown Structure
PDM	Product Data Management
PDW	Product Data Warehouse
PIM	Product Information System
PLCS	Product Life Cycle Support
PLM	Product Lifecycle Management
PPR	Product Process Resource
PROMISE	PROduct lifecycle Management and Information tracking using Smart Embedded Systems
SOM	Semantic Object Model
STEP	Standard for the Exchange of Product model data

TDM	Technical Data Management
UML	Unified Modelling Language
XML	eXtensible Markup Language

1 Introduction

This dissertation aims to improve understanding of knowledge change and will offer ways to cope with such change in the development of knowledge-based applications. The motivation for this work will be discussed first, followed by formulation of a research approach.

1.1 Challenges in Knowledge Engineering for the Aircraft Lifecycle

The study of knowledge has been practiced since Classical times and is known as epistemology. With the advent of the personal computer and associated information technology, the study of knowledge activities such as creation, capture, formalization and implementation has taken flight. Gradually, the field of knowledge engineering has crystallized. This field originated in the early 1980s (Studer *et al.*, 1998) with the specific focus of "integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise" (Feigenbaum and McCorduck, 1983). The use of knowledge engineering for any product lifecycle can be considered by using two perspectives: a theoretical perspective and a practical perspective.

From the theoretical perspective, a central challenge to consider is related to the nature of knowledge and its behaviour over time. What is knowledge and what are its characteristics? How does it behave over time; is it static or does it change? In other words, does knowledge have a lifecycle of its own?

With respect to the issue of knowledge change, current research is rather limited. Schorlemmer *et al.* (2002) discuss the dynamic nature of knowledge: "The dynamic nature of knowledge has long been realised: knowledge evolves over time as experiences accumulate; it is revised and augmented in light of deeper comprehension; entirely new bodies of knowledge are created while at the same time others pass into obsolescence". A similar notion is expressed in Alavi and Leidner (2001) and Nonaka *et al.* (2000). Schreiber *et al.* (1999) state that "…knowledge is not static but changes over time…" and "…knowledge tends to evolve over time" (pg. 184). Stokes (2001) maintains that "knowledge changes rapidly (some talk of a half-life for knowledge of only 20 years)" (pg. 279).

Unfortunately, most authors do not accurately define their concepts, most do not back up their assertions, and none of the mentioned authors go beyond a qualitative assessment of the differences between static (unchanging) and dynamic (changing) knowledge. It seems that our understanding of knowledge is still lacking.

This has major ramifications from a practical perspective. If knowledge changes, existing knowledge-based applications risk becoming rapidly obsolete.

Coenen and Bench-Capon (1993) offer an indication of the magnitude of the problem of knowledge change: the KBS that was studied incorporated an estimated 50% change in rules on a yearly basis, while the overall knowledge base expanded about fourfold in the first 3 years of operation. Van Dijk *et al.* (2012) offers an indication of the costs associated with maintaining a knowledge-based application to keep functionality and knowledge up to date, which are estimated to be 25% of non-recurring software development cost *on a yearly basis*.

So, if understanding of the change of knowledge during its life is lacking, how can existing or to-be-developed knowledge-based applications for engineering tasks cope with knowledge change? All too often, the answer is to (partly) redevelop, to invest in extensive and expensive maintenance, or to abandon the effort altogether. As such, besides achieving a better understanding of the change of knowledge through life, it is necessary to carry through the implications of knowledge change in models and methods to enable the development of more robust engineering applications.

These considerations are of particular interest for the aircraft engineering domain. The development and operation of aircraft are highly complex and collaborative endeavours in which knowledge from numerous stakeholders and disciplines must be integrated to achieve the intended objectives. This holds for the various stages of the aircraft lifecycle, including design, manufacturing and operation support. In tandem with the introduction of new materials (e.g. composites) and production techniques, the continuous development of methods and applications for the various aircraft life cycle stages is a must. The use of knowledge engineering may be employed to support these developments. Knowledge engineering offers methods and models to streamline or even automate engineering processes while retaining the requisite knowledge (Schreiber *et al.*, 1999). This may reduce process time significantly while improving the quality of analysis, decisions and output. However, quite a few challenges remain for the application of knowledge engineering within aircraft engineering (Bermell-Garcia *et al.*, 2012; Verhagen *et al.*, 2012). These challenges focus on *usability* and *maintainability* of knowledge and knowledge-based applications. Knowledge must be geared towards the end user(s), which must be able to retrieve, understand, use and manage the knowledge used in knowledge-based applications. Existing knowledge must be able to be updated following new insights. In other words, knowledge change must be taken into account, allowing for life-cycle management of knowledge and the associated knowledge-based applications.

1.2 Research Approach

Does knowledge change and if so, how is this coped with? These general questions inform the vision and consequently the direction of the research. The

vision of this research is to show that knowledge changes and has a lifecycle which can be modelled and quantified, and to carry through the implications of knowledge change into a set of models and a method to consistently formalize, use and maintain knowledge for engineering tasks within the aircraft lifecycle. To consolidate the stated vision, the following high-level research goal is identified:

> *Support consistent formalization, use and maintenance of changing knowledge within aircraft lifecycle phases to improve domain-specific modelling, execution and control of engineering tasks*

To address the general research challenges discussed in the previous section, a research framework is formulated. This approach consists of the research scope, objectives and research questions (the *what* – Section 1.2.1), as well as the specific research design that outlines the modelling, analysis and validation approach (the *how* – Section 1.2.2). In applying the research design, the theoretical contributions (Chapter 3) and practical contributions (Chapters 4-6) of this dissertation are developed.

1.2.1 Research Framework

The first step in addressing the aforementioned research challenges is to pick up on the general research vision and address *exactly* what it is that is being researched, i.e. the research objectives, scope and questions. These elements are the focus of the following sections.

1.2.1.1 Research Objectives

To achieve the high-level research goal, several research objectives must be met:

1) ***Knowledge life cycle modelling:*** it is necessary to understand and model knowledge through time. Therefore, a model for the lifecycle of knowledge must be developed. This model must enable quantification of knowledge change.
2) ***Ontology-based approach to support knowledge change:*** to support the consistent formalization, use and maintenance of changing aircraft knowledge in its various lifecycles, an ontology-based approach must be developed. The ontology supports the knowledge lifecycle and can be applied during any aircraft lifecycle phase to construct knowledge-based applications that support changing knowledge. The resulting applications must have improved maintainability and usability.
3) ***Methodology development:*** to support application of the ontology-based approach, a methodology must be developed. This should employ the knowledge life cycle model and associated ontology-based approach to

support the development of 'white-box' knowledge-based applications that can handle knowledge change and offer improved maintainability and usability.

To ground the research, the state of the art in related domains is to be explored (see Chapter 2). This review will support the assertions made as part of the research objectives. Furthermore, the proposed models and methodology must be validated. The associated approach is discussed in more detail in Section 3.5.

1.2.1.2 Research Scope

The research is scoped with respect to three aspects.

First, the current research will primarily consider *explicit knowledge*, i.e. knowledge that has been codified and is available in documents and other formalized forms (see also Section 2.1). This choice has been made in order to enable the modelling and quantification of knowledge change. Tacit knowledge – and its conversion into explicit knowledge – is considered as part of the case studies (see also Chapters 4-6).

Second, instead of considering a generic product lifecycle, the research focuses on the aerospace domain; the *aircraft lifecycle* will be studied. In particular, the design, manufacturing and maintenance phases of the aircraft lifecycle are considered. These phases are most directly associated with the generation, formalization and (re-)use of explicit knowledge in knowledge-based applications. Therefore, they are the most suitable phases for further research. A final note regarding these phases is that the emphasis lies on *case study research* of *engineering tasks* as encountered in the design, manufacturing and maintenance phases of the lifecycle.

Two of three case studies will be concerned with *thermoset composite products*. This is an area of considerable interest in both the research and the business communities, given the introduction of the Boeing B787 and Airbus A350XWB and the associated required developments in design, manufacturing and maintenance processes.

To summarize, the research has been scoped to address *knowledge* within *aircraft lifecycle phases (design, manufacturing, maintenance)* with a particular interest in *thermoset composite products*.

The following aspects will *not* be included into the research scope:

- **Knowledge exchange across aircraft lifecycle stages:** knowledge is generated during various stages of the aircraft lifecycle. Some knowledge may originate in early lifecycle stages (e.g. design) and move through subsequent stages (e.g. manufacturing, maintenance). The change of knowledge over these stages will not be addressed in this dissertation.

- **Application interoperability across the aircraft lifecycle:** the aforementioned use case research focuses on knowledge-based application development for individual life cycle stages. The interoperability of applications across life cycle stages (e.g. a design tool interacting with a maintenance tool) will not be considered.
- **Organizational factors:** knowledge-based applications do not exist in a vacuum. Organizational factors play an important role in the development, implementation and maintenance of knowledge bases and applications, but these factors are not considered in detail in the current research.
- **Automatic translation between informal and formal knowledge representations:** knowledge can be collected using informal and formal representations (see Section 2.2.3), which are related to each other. Typically, an informal representation of knowledge is the first step in a process leading to formalization of knowledge, which consists of the modelling and implementation of knowledge in knowledge-based applications. One of the most appealing research challenges in knowledge engineering is to make it possible to automatically link and convert informal to formal knowledge. Automatic translation models and mechanisms need to be developed. This would open up the path to rapid knowledge-based application development, while improving maintainability and usability of knowledge. However, this challenge is not addressed in the current research.
- **Task automation:** while developing and implementing knowledge-based systems, it is typical to automate repetitive tasks, especially in KBE development. Though it may feature in some case studies, automation is in itself not a research objective for this dissertation.

1.2.1.3 Research Questions

A number of research questions are formulated to direct the research. With respect to the theoretical challenge and the related research objective – knowledge lifecycle modelling – the following questions are considered:

- Which concepts and relationships are required to characterise the change of explicit knowledge within and throughout the aircraft lifecycle phases?
- How does explicit knowledge change within specific phases of the aircraft lifecycle?
- Is change of explicit knowledge quantifiable?

These questions will be partially answered in Section 3.1, where a knowledge life cycle model is proposed. Relative to the model, two general hypotheses (and associated null hypotheses) are introduced here.

H_1: The frequency of knowledge actions decreases along the knowledge lifecycle
$H_{0,1}$: The frequency of knowledge actions remains equal or increases along the knowledge lifecycle
H_2: Number of knowledge actions per year increases during the aircraft lifecycle
$H_{0,2}$: Number of knowledge actions per year remains equal or increases during the aircraft lifecycle

These hypotheses are further explained and tested in Section 6.2.2, along with a set of case-specific hypotheses. The mentioned Section also answers the remaining research questions regarding knowledge lifecycle modelling.

With respect to the practical challenges and the related research objectives – the development of an ontology-based approach and supporting methodology – the following questions are considered:

- Which concepts and mechanisms support the consistent formalization, use and maintenance of changing knowledge throughout the aircraft lifecycle?
- How can knowledge change be accommodated during knowledge-based application development?
    - Which models are required and how do these models help to accommodate knowledge change?
    - Which steps are required?

1.2.2 Research Design

To find answers to the research questions and meet the research objectives, a three-stage research design has been adopted. Figure 1.1 presents the resulting research framework, which consists of identification of state-of-the-art and shortcomings through literature review, development of contributions through theory, which are validated through practical application in three case studies.

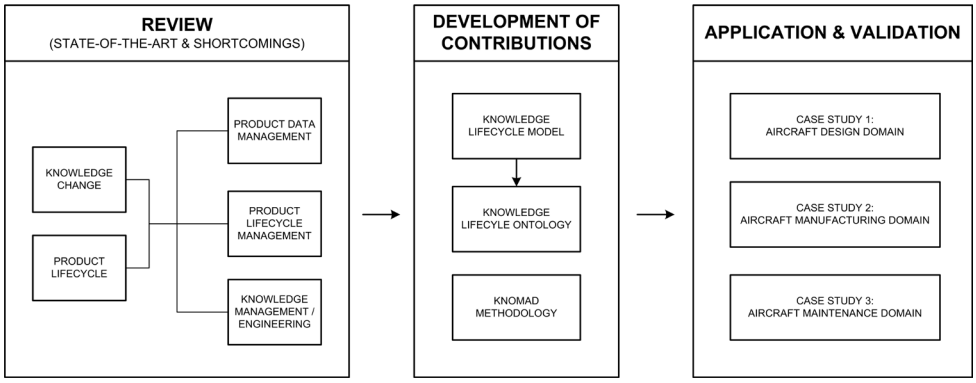

Figure 1.1: Research framework

The two-stage research design is further explained below.

1.2.2.1 Theory Development: Research Contributions

Theory development consists of three specific contributions that tie in with the research objectives:

- The first contribution is an effort to conceptualize and model the behaviour of knowledge over the aircraft lifecycle (Chapter 3.1).
- The second contribution to theory is the development of an ontology for the support of knowledge change in knowledge-based applications (Chapter 3.2). It combines elements of the knowledge lifecycle model with previous work in the PLM and KBE research domains. The ontology can be applied in the development of knowledge-based applications.
- The third contribution is the development of the KNOMAD methodology, supporting the application of an ontology-based approach in the development of knowledge-based applications that have to cope with knowledge change (Chapter 3.3).

1.2.2.2 Practical application: Implementation and Validation

The developed models and methodology are put to the test in three case studies for specific aircraft life cycle phases – design, manufacturing and maintenance. For each case study, the research context and challenges are indicated. The theory contributions are then applied to the particular case: knowledge change is characterised using the knowledge lifecycle model, task analysis is performed to support ontology application, and the KNOMAD methodology is applied to the case to produce a development flow chart. This results in development of a knowledge-based solution for the research problem. The three case studies are presented in Chapters 4-6.

For the maintenance phase, the knowledge lifecycle model is additionally tested and validated by quantitative, statistical analysis of the behaviour of

knowledge. The model concepts are operationalized and an appropriate representation of knowledge is selected. Following this, general and case-specific hypotheses are posited and two separate research samples are gathered, processed and statistically analysed.

1.3 Dissertation Structure

The structure of this dissertation is informed by the research design. Figure 1.2 gives the research roadmap. It shows that the research context will be explored first; this includes a discussion of the state-of-the-art in relevant research domains. This is followed by the development of theoretical and practical contributions. Finally, conclusions are drawn and the research is discussed. The individual chapters are discussed in more detail below.

Chapter 2 (Exploration of the Research Context) gives an overview of research in various fields that are related to the problem statement as given in this introduction. First, accurate definitions of knowledge and knowledge change are sought, particularly in relation with data and information. The product lifecycle concept is introduced next. The concepts of data, information and knowledge are then extended across the product lifecycle, which leads to review of state-of-the-art and shortcomings in the fields of Product Data Management, Product Lifecycle Management and Knowledge Management and Engineering. After discussion of these research fields, a number of research challenges are defined.

Chapter 3 (Theory Development) presents the three major contributions to theory of this dissertation. The first section introduces a conceptual knowledge lifecycle model that aims to enable the characterisation and quantification of knowledge change. The second section of Chapter 3 introduces the model that can be used to support the consistent formalization, use and maintenance of knowledge within aircraft lifecycle phases. This model is an ontology, a representation of the concepts and relationships in a domain (Uschold and Gruninger, 1996; Noy and McGuinness, 2009), and combines the knowledge life cycle concept and its attributes with elements from research on lifecycle ontologies and functional modelling. The third section establishes a methodology that can be used to support consistent formalization, use and maintenance of knowledge over the aircraft lifecycle. It contains a number of distinct steps that can be used to develop knowledge-based applications that can cope with knowledge change. The contributions are discussed in Section 3.4. The final section of Chapter 3 outlines the approach to validate the contributions to theory by introducing the case study approach.

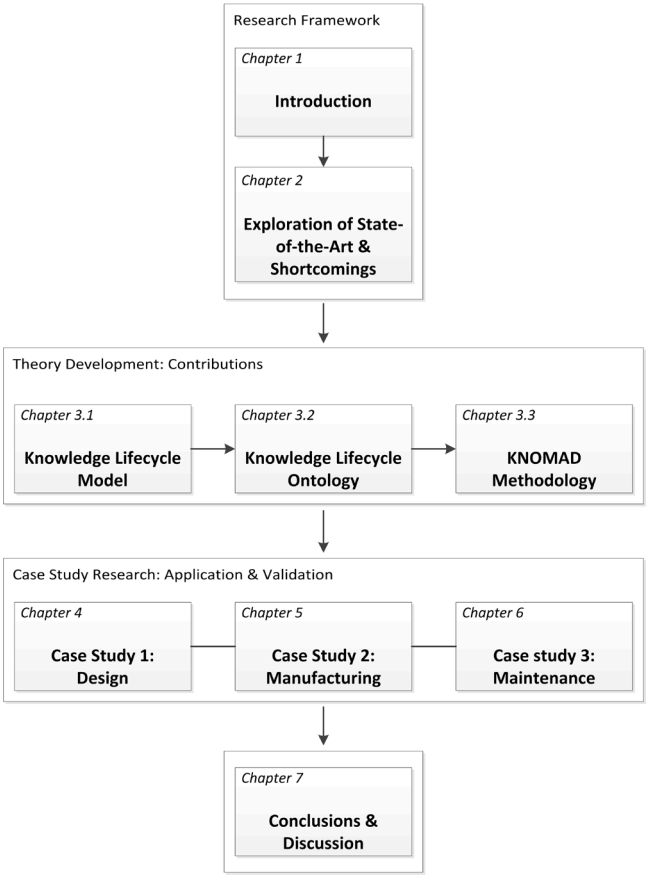

Figure 1.2: Research roadmap

Chapter 4 (Case Study 1: Aircraft Design for Manufacturing) discusses the development of a knowledge-based application to optimize the design of a thermoset composite wing panel for ply continuity, which enables better manufacturability. The theoretical contributions introduced in Chapter 3 are applied to the case study by identification of knowledge change, engineering task analysis and application of the methodology. Subsequently, a knowledge-based application is developed through application of the methodology steps and the knowledge lifecycle ontology. The knowledge-based application meets requirements with respect to usability and maintainability, as well as specific case study requirements.

Chapter 5 (Case Study 2: Aircraft Manufacturing Cost Modelling) details the development of a knowledge-based application for cost modelling and estimation for the manufacturing of a thermoset composite wing. As in Chapter 4, the theoretical contributions introduced in Chapter 3 are applied to the case study by

identification of knowledge change, engineering task analysis and application of the methodology. Subsequently, a knowledge-based application is developed through application of the methodology steps and the knowledge lifecycle ontology. The knowledge-based application meets requirements with respect to usability and maintainability, as well as specific case study requirements, though the application for this use case is semi-automatic and requires user interaction to function.

Chapter 6 (Case Study 3: Aircraft Maintenance Modelling) discusses the development of a maintenance knowledge base that captures and formalizes knowledge for a specific maintenance function: the inspection and modification of a Boeing B737 slat main track downstop assembly. As such, it includes lifecycle knowledge coming from external sources (e.g. the Original Equipment Manufacturer, Boeing, and the regulator, the Federal Aviation Authority). As before, the theoretical contributions introduced in Chapter 3 are applied to the case study by identification *and* quantification of knowledge change, as well as engineering task analysis and application of the methodology. Subsequently, a knowledge-based application is developed through application of the methodology steps and the knowledge lifecycle ontology. The knowledge-based application meets requirements with respect to usability and maintainability, as well as specific case study requirements.

Chapter 7 (Conclusions & Discussion) first synthesizes the contributions from the case studies in light of the developed theory. The research objectives and questions of this dissertation are then revisited; appropriate conclusions are drawn. In the Discussion section, the contributions and limitations of the performed research are discussed. Furthermore, recommendations for future research are given.

Appendix A (Complexity Estimation) includes a brief discussion of complexity estimation for the development of knowledge-based applications consisting of modular elements.

2 Exploration of the Research Context

To gain a better perspective on the aspects of the expressed research vision, the state-of-the-art of the related research fields is described. Furthermore, the high-level research challenges identified in the introduction will be refined. This will result in specific, motivated research challenges as a basis for further research.

The first aspects that will be explored are the definitions of knowledge and knowledge change. Which perspectives exist and which definitions are adopted in this dissertation? Following the definition of these concepts, generic representations of the product lifecycle are discussed, with further specification in terms of the aircraft lifecycle. Applying a knowledge perspective to the product lifecycle gives rise to discussion of the research fields of Product Data Management (PDM), Product Lifecycle Management (PLM), Knowledge Management (KM) and Knowledge Engineering (KE).

2.1 Perspectives on Knowledge and Knowledge Change

What is knowledge? What is its nature; how can it be defined? And does it change, and if so, how can this be defined?

From the perspective of this dissertation, it would go too far to consider all philosophical angles on knowledge. Rather, the focus will be on two major and often used perspectives to define knowledge, as discussed at length by Alavi and Leidner (2001):

1) The notion of a hierarchy from data to information to knowledge.
2) Knowledge taxonomies, in particular the distinction between explicit and tacit knowledge.

As Hicks *et al.* (2002) note, the words data, information and knowledge are often used in an interchangeable manner by practitioners, which "complicates the identification and development of mechanisms for the capture, storage and reuse of each resource". As the interchangeable use by practitioners indicates, the notions of data, information and knowledge are closely related. This realisation has brought forth a major and often used perspective on knowledge: the notion of a hierarchy from data to information to knowledge (Wiig, 1997; Nonaka *et al.*, 2000; Alavi and Leidner, 2001; Bufardi *et al.*, 2005; Ouertani *et al.*, 2011), where data precedes information, which in turn precedes knowledge. It is implicitly assumed that value increases intrinsically from data to information to knowledge. In contrast, researchers such as Tuomi (1999) argue for a reversed hierarchy, where the existence of knowledge precedes the existence of information and data. In this dissertation, no judgement will be given with regards to the inherent

value and supposed 'progress' from data to information and to knowledge, or vice versa. Instead, the focus lies on obtaining definitions of these three related concepts that are as clear as possible. The three central concepts of data, information and knowledge are discussed below:

- **Data:** data can be considered as “raw numbers and facts” (Alavi and Leidner, 2001), “simple facts” (Tuomi, 1999), “symbols which have not yet been interpreted” (Van der Spek and Spijkervet, 1997), “simple observations of states of the world” (Davenport and Prusak, 1998), or “unorganized and unprocessed facts” (Ameri and Dutta, 2005). Hicks *et al.* (2002) offer a slightly more involved discussion of the concept of data, including a look at the difference between structured and unstructured data, and noting that the 'facts' alluded to in the definitions of others indicate occurrences of a measure or inference of some quantity or quality. Finally it should be noted that some authors (e.g. Simon *et al.* (2001)) make a distinction between static data (specification of the product, i.e. data that is created once and stays intact during the product lifecycle) and dynamic data (data collected during the use of a product).
- **Information:** like data, information is defined in different ways. For instance, Tuomi (1999) defines information as structured simple facts. Van der Spek and Spijkervet (1997) define information as data with meaning, whereas Alavi and Leidner (2001) and Tuomi (1999) maintain that information is meaningless in itself; for them, meaning is the defining characteristic that transforms information into knowledge. Tuomi (1999) adds that “the general accepted view sees data as simple facts that become information as data is combined into meaningful structures”. Davenport and Prusak (1998) sees information as “data endowed with relevance”. Wiig (1997) states that information “consists of facts and data that are organised to describe a particular situation or condition”. Hicks *et al.* (2002) and Ouertani *et al.* (2011) combine some of the previous perspectives by expressing information as having two aspects: “…a subject or descriptor, which provides the meaning, and a predicate or value that holds the measure, typically a data element” (Hicks *et al.*, 2002). From this perspective, information can be expressed as data within a context. Furthermore, Hicks *et al.* (2002) consider the difference between informal and formal information. Informal information is seen as unstructured information; information possessed by individuals where subjects and predicates are not clearly defined and may change dynamically. Formal information is an element of information possessing a specific context and measure; its must be structured and sufficiently decomposed to act as a platform to infer knowledge from.

- **Knowledge:** the concepts of information and knowledge are often used interchangeably. However, it is possible to establish some essential differences between these two concepts. A number of authors focus on the inclusion of (personalized) meaning as the defining difference. For instance, Van der Spek and Spijkervet (1997) state that "knowledge is what enables people to assign meaning and thereby generate information". Tuomi (1999) maintains that "information, in turn, becomes knowledge when it is interpreted, put into context, or when meaning is added to it". Wiig (1997) defines knowledge as follows: "knowledge consists of truths and beliefs, perspectives and concepts, judgments and expectations, methodologies and know-how". Finally, Alavi and Leidner (2001) state that "what is key to effectively distinguishing between information and knowledge is not found in the content, structure, accuracy or utility of the supposed information or knowledge. Rather, knowledge is information possessed in the mind of individuals". Alavi and Leidner (2001) furthermore maintain that only information that is actively processed in the mind of an individual or individual(s) is useful.

  The aspects of usability and applicability inform another dominant stream of definitions for knowledge. For instance, a hint of these aspects is included in the definition from Tuomi (1999) who states that "the general accepted view sees data as simple facts that become information as data is combined into meaningful structures, which subsequently become knowledge as meaningful information is put into a context and when it can be used to make predictions". Usability and applicability are much more explicitly considered in the following definitions. First, Ouertani *et al.* (2011) focus solely on usability and applicability as defining aspects of knowledge over information: "Knowledge on the other hand is information with added details relating how it should be used or applied". Nonaka (1994) uses the classic epistemological definition of knowledge and adds a consideration regarding usability by defining knowledge as "a justified true belief that increases an entity's capacity for effective action". Ameri and Dutta (2005) see knowledge as "evaluated and organized information that can be used purposefully in a problem solving process". Gielingh (2005) maintains that "Knowledge is a structure of associations between memorized experiences that enables a human being to perform a task". Schreiber *et al.* (1999) state that "knowledge is the whole body of data and information that people bring to bear to practical *use in action*", where "knowledge adds two distinct aspects: first a sense of purpose…second, a generative capability".

Besides defining the concepts and hierarchical interpretation of data, information and knowledge, there is another major perspective on knowledge that will be considered within the context of this thesis: knowledge taxonomies. A significant number of knowledge taxonomies exist (Alavi and Leidner, 2001); the most relevant of these are discussed.

Hicks *et al.* (2002) distinguish between two fundamental elements of knowledge: the object and the process. The view of knowledge as an object is relatively common-place and sees knowledge as a thing that can be stored and manipulated (McQueen, 1998; Alavi and Leidner, 2001). Knowledge elements are inferred from information elements using knowledge processes. A knowledge process is "the procedure(s) utilised by the individual to infer the knowledge element from information, other knowledge elements or a combination of each. These knowledge processes are generally within-person processes". The concept of a knowledge process is further deepened by Nonaka *et al.* (2000), who posit the *SECI* (Socialisation, Externalisation, Combination, Internalization) model to describe the creation of knowledge. The *SECI* model is built upon the realization that knowledge can be created and transferred by the interaction between two types of knowledge: tacit and explicit knowledge.

These dimensions are the constituent parts of another fundamental knowledge taxonomy, which distinguishes between the tacit and explicit dimensions of knowledge (Polanyi, 1966; Nonaka, 1994). The tacit dimension of knowledge, also called tacit knowledge in short, is comprised of a cognitive element, which refers to an "individual's mental models consisting of mental maps, beliefs, paradigms, and viewpoints" (Nonaka, 1994; Alavi and Leidner, 2001) and a technical element, consisting of "concrete know-how, crafts, and skills that apply to a specific context"(Nonaka, 1994; Alavi and Leidner, 2001). The explicit dimension of knowledge, also simply termed explicit knowledge, is defined as being "articulated, codified, and communicated in symbolic form and/or natural language" (Nonaka, 1994; Alavi and Leidner, 2001). The explicit dimension is much more suitable for storing and manipulating of knowledge, and as such relates closely to the view of knowledge as an object. Explicit knowledge is an essential part of the scope of this dissertation (see Section 1.2.1.1), whereas tacit knowledge plays a secondary role in the case studies.

Based on the preceding discussion, throughout this dissertation the following definitions for data, information and knowledge are used:

- **Data:** *data is considered to represent an occurrence of a measure, such as a quantity, which represents an observation and/or fact*.
- **Information:** *data within a structured context: a combination of predicate(s) or value(s) that hold the measure(s), and contextual descriptor(s) that enable structural representation.*

- **Knowledge:** *processed information resulting in a capability for effective action.*

Figure 2.1 illustrates these concepts and their interactions. The transformations from data to information to knowledge and vice versa are shown, as well as the ultimate result of the application of knowledge: an action. Actions can in turn generate new data, information and knowledge.

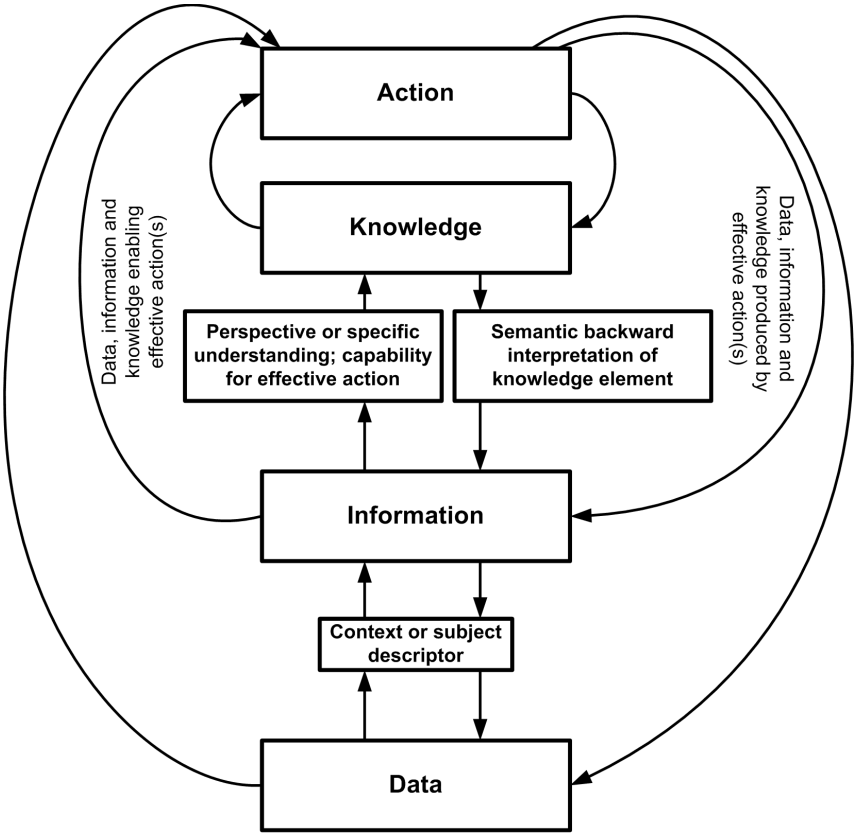

Figure 2.1: Data, information and knowledge transformation processes (adapted from Hicks *et al.* (2002))

This definition of knowledge can be extended to define knowledge change:

Knowledge change: *change in knowledge over time, where knowledge is defined as processed information with a capability for effective action*.

Change as a concept incorporates not only an alternation of an existing element, but also includes addition or exclusion of an element and its constituent parts.

Change of knowledge can be typified from simple to complex, associated with the data-information-knowledge hierarchy. For simple change (**data change**), the values associated with knowledge elements alter from time t_1 to time t_2. Attribute values may change as well as values used in rules or predicates. A more involved form of change is associated with information (**information change**): the structured context of a knowledge element changes. The type, number and applicability of relations for a specific knowledge element may vary over time. Finally, on a knowledge level, the capability for effective action associated with a knowledge element can change (**knowledge change**). This can be caused by changes in rules (change, addition or exclusion of antecedents and/or consequents), logic (change / revision, addition or invalidation / exclusion of propositions or predicates) or attribute sets (change, addition or exclusion of attributes), depending on the formalism chosen to represent knowledge. Knowledge change may close or open options to achieve effective actions.

To illustrate the various types of change, a short example is discussed. The example considers a simple engineering task: selection of a material based on requirements relative to material properties and cost. A diagram is shown in Figure 2.2 to explain this task and the constituent data, information and knowledge. In this Figure, one sees three classes: **Material**, **Metal_Material** and **Material_Requirements**. The first class (**Material**) has attributes[*] *E_modulus*, *G_modulus* and *Cost*, representing Young's modulus *E* (in GigaPascal, GPa), the shear modulus *G* (GPa) and a hypothetical cost *C* in dollars per m^3 of material. The **Metal_Material** class is a subclass of **Material** and inherits the attributes. The **Material_Requirements** class has attributes to express the requirements on the material attributes: *Required_E_modulus*, *Required_G_modulus*, *Required_Cost*. In addition to the classes, two objects have been instantiated to represent two different metal alloys: Al2024T3 (aluminium alloy) and Ti6A14V (titanium alloy). The mechanical properties are taken from Baker *et al.* (2004). The cost figures are hypothetical. Another object has been instantiated to represent requirements for the material selection. A final element, which is not represented in the Figure, is the rule (set) that can be used to select a material that meets the requirements. In natural language, this can be expressed in the following way: if the E_modulus of a material is larger than the required E_modulus *and* if the G_modulus is larger than the required G_modulus *and* if the material cost is lower than the required material cost, then select the material with the lowest cost. If any of the requirements is violated, the material is rejected for selection outright. If multiple materials meet the requirements, a simple method can be written to select the

[*] Note that the class attributes can be considered to be highly incomplete for a true representation of material properties; a limited number of properties is included as the objective of the example is to illustrate the types of knowledge change that may occur.

material with the lowest cost. A more formal notation of the selection rule is given in Table 2.1.

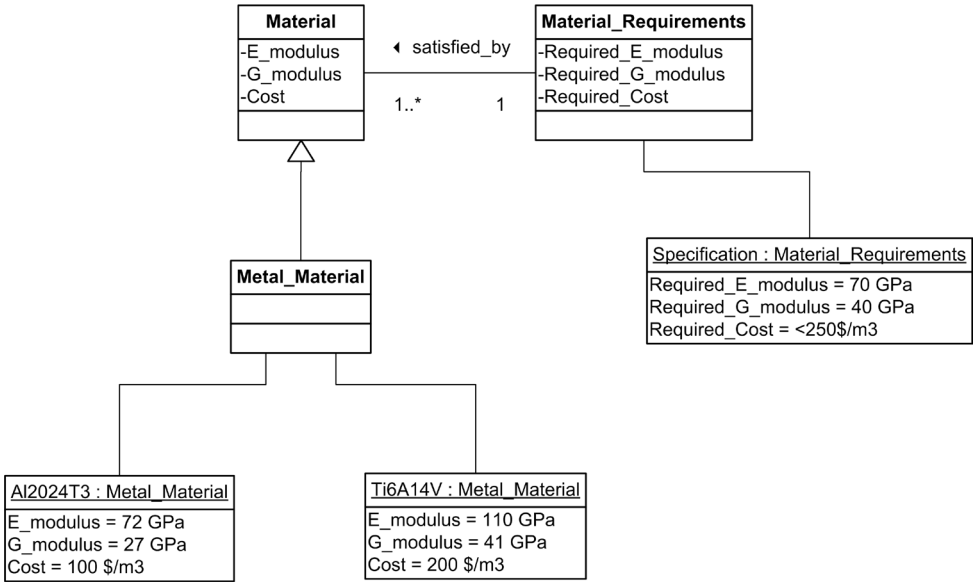

Figure 2.2: Selection of a material – baseline state

Consider Figure 2.3, which shows a change in state with respect to the baseline. Numerous changes have been incorporated; these are highlighted in red.

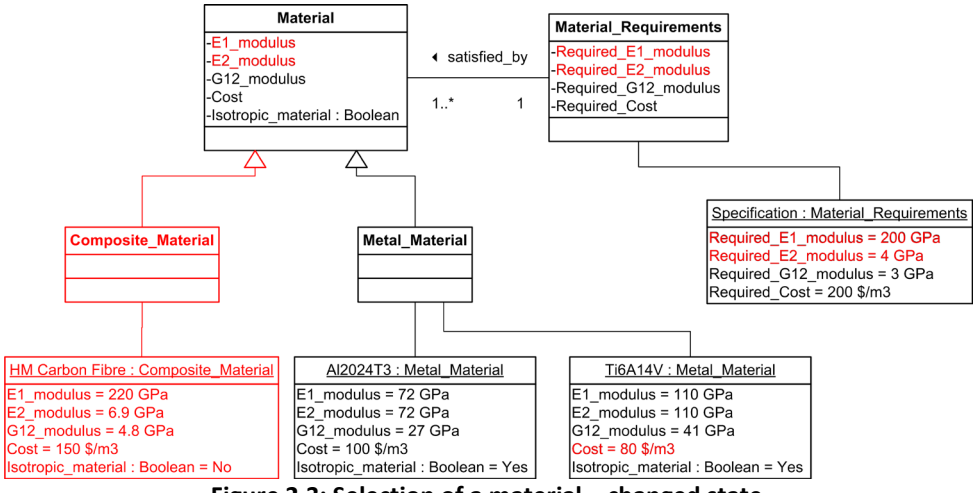

Figure 2.3: Selection of a material – changed state

Table 2.1 summarizes the type of changes that have been incorporated. These changes are consistent with the definitions for the types of change given above:

- Data change is represented by a change in attribute value for an object ($C_{Ti6A14V}$): the outcome of selection may change entirely, but the change itself is simple;
- Information change is represented by a change in structured context (additional class **Composite_Material** and associated parent-child relation, as well as an additional boolean property *isotropic_material* as this drives material properties), leading to a new option for a material that can meet the given requirements;
- Knowledge change is represented by a change in the *E_modulus* attribute, which has been split up into a longitudinal E-modulus E_1 and transversal E-modulus E_2 to enable the representation of the anisotropic material properties of composite materials – the knowledge is *revised*. The rule associated with the selection of a material has changed accordingly to include antecedent criteria with respect to E_1 and E_2. In addition, a rule is added to decompose the properties of isotropic materials into longitudinal and transversal representation – knowledge is *expanded*.

Table 2.1: Examples of data, information and knowledge change and implications

Type of change	Original state	Changed state	Outcome
Data change	$C_{Ti6A14V}$ = 200	$C_{Ti6A14V}$ = 80	Ti6A14V[†]
Information change	Al2024-T3 Ti6A14V	Al2024-T3 Ti6A14V High-Modulus (HM) Carbon Fibre composite	HM CF
Knowledge change (revision)	IF $(E_{mat} \geq E_{req}) \wedge (G_{mat} \geq G_{req})$ $\wedge (C_{mat} \leq C_{req})$ THEN $Select(Mat(C_{\min}))$ ELSE IF $(E_{mat} \leq E_{req}) \vee (G_{mat} \leq G_{req})$ $\vee (C_{mat} \geq C_{req})$ THEN $reject(Mat)$	IF $(E_{1,mat} \geq E_{1,req}) \wedge (E_{2,mat} \geq E_{2,req})$ $\wedge (G_{12,mat} \geq G_{12,req}) \wedge (C_{mat} \leq C_{req})$ THEN $Select(Mat(C_{\min}))$ ELSE IF $(E_{1,mat} \leq E_{1,req}) \wedge (E_{2,mat} \leq E_{2,req})$ $\wedge (G_{12,mat} \leq G_{12,req}) \wedge (C_{mat} \geq C_{req})$ THEN $reject(Mat)$	HM CF
Knowledge change (expansion)	-	IF $Isotropic\_material = True$ THEN $E_{mat} = E_{1,mat} = E_{2,mat}$	$E_{Al2024T3}$ = $E_{1,Al2024T3}$ = $E_{2,Al2024T3}$ = $E_{Ti6A14V}$ = $E_{1,Ti6A14V}$ = $E_{2,Ti6A14V}$

The use of the terms *revision* and *expansion* for knowledge change is not coincidental. These terms are co-opted from literature on belief revision (Doyle, 1979; Martins and Shapiro, 1988; Kern-Isberner, 2004), which uses logic as a basis to specify theorems and proofs for revising knowledge base content given the introduction of new and possibly contradicting beliefs (Martins and Shapiro, 1988). In the case of conventional reasoning, monotonicity applies: 'beliefs are true, truths never change, so the only action of reasoning is to augment the set of

[†] If data change is the only change that occurs, selection of Ti6A14V is the logical outcome as only the two metals are considered. If information and/or knowledge change occurs, the selection outcome will of course be different.

beliefs with more beliefs' (Doyle, 1979). To enable revision of beliefs, non-monotonic logic must be considered. Non-monotological systems are 'logics in which the introduction of new axioms can invalidate old theorems' (McDermott and Doyle, 1980). To enable belief revision, several types of truth maintenance systems or TMS (also referred to as belief revision systems) have been developed (Doyle, 1979; de Kleer, 1986; Martins, 1990).

TMS incorporating non-monotonic logic are typically expressed using propositional or predicate logic formalisms (Martins and Shapiro, 1988; Katsuno and Mendelzon, 1991). As such, with respect to the issue of knowledge change, any knowledge base that uses such formalisms might be evaluated for change. This could conceivably be achieved by using propositional or predicate logic to represent knowledge elements in a knowledge base application, and subsequently test knowledge base use over time. However, to the best of the author's knowledge, empirical studies to perform quantification of knowledge change using truth maintenance systems have never been performed. A possible explanation is that knowledge bases tend to be expressed either in informal, rule-based (IF-THEN) or frame-based ways (La Rocca, 2012).

To summarize, the concept of knowledge has been defined using the data-information-knowledge hierarchy. Subsequently, various modes of change have been discussed, culminating in knowledge change. With respect to the latter, a shortcoming of current literature is a lack of quantification of knowledge change. This finding is further considered in Sections 2.2.3.1 and 2.3.

As a result of this consideration of knowledge change, a contribution to theory is developed in Section 3.1 relative to conceptual modelling of the lifecycle of knowledge. This enables characterisation and quantification of knowledge change. In the case studies (Sections 4-6), this contribution to theory is validated through qualitative and quantitative means, which is explained in more detail in Section 3.5.

2.2 State of the Art and Challenges for Knowledge Perspectives along the Product Lifecycle

As noted in the introduction, the product lifecycle – and more specifically, the aircraft lifecycle – is an essential part of the scope of this thesis. As such, this concept will first be briefly defined. The product lifecycle has been discussed extensively in literature (e.g. Aitken *et al.* (2003), Brissaud and Tichkiewitch (2001), Krozer (2008) and Thimm *et al.* (2006)). Broadly speaking, two notions regarding the product lifecycle exist: a market-oriented, commodity view of the product lifecycle (Aitken *et al.*, 2003) and a more process-oriented lifecycle view (Brissaud and Tichkiewitch, 2001; Thimm *et al.*, 2006; Krozer, 2008). The first notion of a product lifecycle typically contains product introduction, growth,

maturity, saturation and decline as major stages, with some minor variants. The second conceptualization of the product lifecycle is process-oriented. This view is of primary interest within the scope of this thesis and can be defined as follows:

- **Product design:** the first phase in the product lifecycle, product design is about the development of a product that has to meet desired needs. As such, it encompasses the creation and analysis of the (geometric) description of a thing to be built (Raymer, 2006), based upon the inputs of customers in the form of requirements, physical or stakeholder constraints and supported by analysis of performance and functionality. Product design typically relies on and synthesizes knowledge from various analytical disciplines. For instance, in aerospace design, disciplines such as aerodynamics, structures, controls and propulsion all feed into the design process. In aerospace, the product design phase is typically broken down in three major phases (Raymer, 2006): conceptual design (characterised by a large number of design alternatives and trade studies, and a continuous change to the aircraft concepts under consideration), preliminary design (characterized by a maturation of the selected design approach) and detail design (characterized by a large number of designers preparing detailed drawings with actual fabrication geometries and dimensions). The design phase can encompass the production of prototypes and subsequent testing.
- **Product manufacturing:** following the finalized design of the product, it will have to be produced. Manufacturing is about the transformation of raw materials into a finished product (Mazumdar, 2002), based on the product design specifications. Typically, production or manufacturing of a product is preceded by the design of the manufacturing system, followed by production planning, pilot-scale production, full-scale production, inspection of completed products, and finally distribution towards the customer. Manufacturing processes can be characterised in multiple ways, for instance by production rate, cost, performance, size and shape (Mazumdar, 2002).
- **Product operation:** Product operation refers to the actual use of a product. In the aerospace sector, this essentially comes down to flying an aircraft for its intended purpose, be it passenger or cargo transport, or military operations, or any other function. Though aircraft operators are often identified with large airline companies such as Air France-KLM, operators can be individuals, government branches, charter companies, and others.
- **Product support:** to ensure the proper functioning of a product, it is often necessary to support it during its operational life. This can include

maintenance and repair of a product to prevent or fix any operational issues and to keep it in regular working order (i.e. a state in which it can perform its required function (European Federation of National Maintenance Societies, 2011)). In the aerospace sector, product support is typically known as Maintenance, Repair and Overhaul (MRO) and can be considered as a subsidiary stage in aircraft operations. MRO encompasses all forms of maintenance (corrective, preventive and predictive (Jun *et al.*, 2007)), consequently encapsulating both scheduled and unscheduled forms of maintenance and repair.

- **Product disposal:** at end of life, a product enters the disposal stage in which it can either be (partially) re-used, remanufactured or recycled (Jun *et al.*, 2007) – for instance by adherence with the Cradle-to-Cradle philosophy (McDonough and Braungart, 2002) – or in which it is reduced to waste. In the aerospace sector, aircraft disposal comes in different guises: selling an aircraft to another operator (frequently in developing countries), handing the aircraft back to the lease company, storing an aircraft for future use or dismantling an aircraft while retaining useful parts for future use as part of the spare parts pool.

A number of minor variants for this process-oriented lifecycle perspective can be identified, such as the separation of market requirements and conceptualization prior to the product design stage (Thimm *et al.*, 2006), and raw materials supply as a separate stage rather than contributing elements in the manufacturing stage (Thimm *et al.*, 2006; Krozer, 2008).

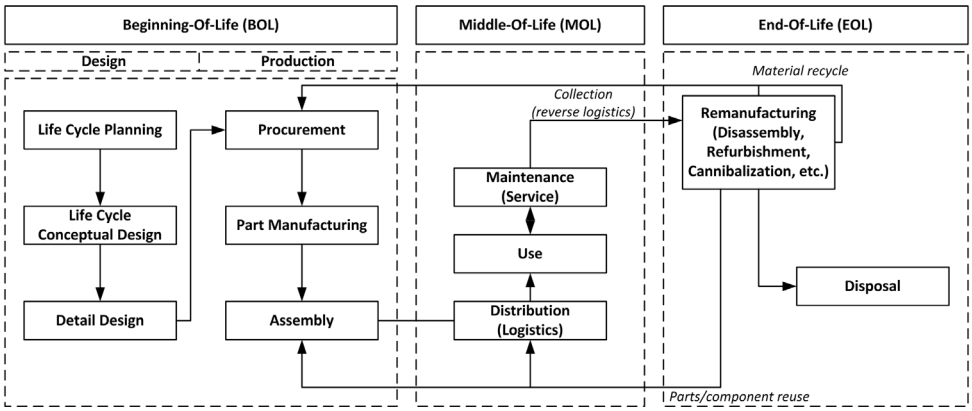

Figure 2.4: Product lifecycle stages (adapted from Jun *et al.* (2007))

Figure 2.4 shows an alternative classification frequently used in research, which identifies a beginning-of-life (BOL) stage including design and production, a middle-of-life (MOL) stage including logistics, use, service and maintenance, and

an end-of-life (EOL) stage, including reverse logistics, remanufacturing, reuse, recycle and disposal (Jun *et al.*, 2007).

For the remainder of this section, the specific focus is on knowledge engineering through product life. The previously introduced data-information-knowledge hierarchy will be extended across the product lifecycle as a structure or guide for the subsequent discussion of three specific research domains: Product Data Management (PDM), Product Lifecycle Management (PLM) and Knowledge Management, the latter of which comprises several knowledge-related fields of study. Figure 2.5 shows the application of this thought. First, data through product life is encapsulated in Product Data Management (PDM). Information through product life is considered within the field of Product Lifecycle Management (PLM) and knowledge through product life is connected with Knowledge Management. Each of these domains is discussed in terms of the state-of-the-art (historical roots, definition, functionality and benefits) and shortcomings.

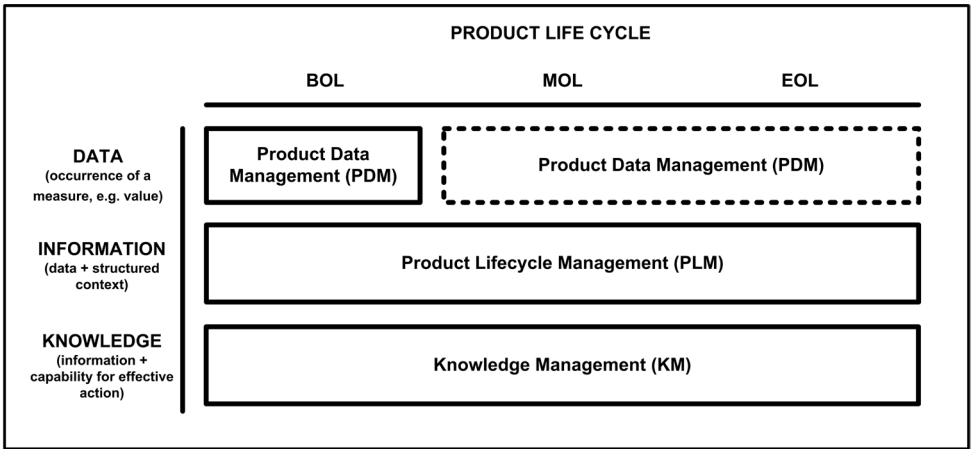

Figure 2.5: Research domains of interest

2.2.1 Data through Product Life: Product Data Management

With the advent of computer-based engineering tools such as Computer Aided Design (CAD), Computer Aided Manufacturing (CAM) and Computer Aided Engineering (CAE) in the early 1980s, large corporations were faced with some pressing issues: how would they manage their data digitally instead of on paper? How would data from many – often disparate – sources be managed? And given the contention that data can change dynamically over the lifecycle (Simon *et al.*, 2001), how should data be managed over the lifecycle of a product? To tackle these challenges, large corporations in the early 1980s developed their own data management solutions (Liu and Xu, 2001). In the meantime, a research community coalesced around the extension of existing techniques such as

engineering data management (EDM), document management, product information management (PIM) and technical data management (TDM) into a new field of research: Product Data Management (PDM), a "common term, encompassing all systems that are used to manage product information" (Philpotts, 1996). In the late 1980s, the first generation of commercial PDM systems had arrived (Liu and Xu, 2001).

PDM can be defined as *a set of tools that help engineers in managing the data and the processes related to the product development life cycle* (Philpotts, 1996; Bilgic and Rock, 1997). As such, its main focus is on the design and manufacturing stages of the product life cycle. However, as Liu and Xu (2001) note, PDM can also be extended to end-user support, in effect increasing its range towards the operation and product support stages of the product lifecycle. A number of substantial benefits are claimed for PDM systems, including interdisciplinary collaboration, reduced product development time, reduced complexity of accessing information, improved project management and improved lifecycle design (Liu and Xu, 2001), as well as access to the most up to date information and productivity gains (Philpotts, 1996).

To achieve the promised functionality and benefits, PDM systems are typically comprised of three elements: an electronic vault or data repository, a set of user functions and a set of utility functions (Philpotts, 1996). The electronic vault contains product data (such as specifications, CAD models, CAE data, and maintenance records) and meta-data to support PDM system functionality. Meta-data is included to store information about product data by descriptive attributes such that changes, release levels, authorizations and other data controls can be tracked and audited. Furthermore, the inclusion of meta-data (i.e., data about data, such as authorship, time of creation, etc.) can also be used to create relationships among product data such that data context can be created, which is an important step towards the creation of information. The user functions provide the interface to PDM functionalities such as data storage, retrieval and management. Important user functions are document management, product structure management (in which product data is organized and stored), workflow and process management (which assists in sending the right available data at the right time to the right user), classification of data by attributes, and programme management (Philpotts, 1996; Eynard *et al.*, 2006). Finally, utility functions are the third essential part of PDM systems: these provide the interface with the operating environment. Examples include communication and notification, data transport and translation, and image services. A more involved discussion of PDM, as well as a comparison of commercial PDM systems on a range of functionalities, is available in Bilgic and Rock (1997).

Years of research work and commercial use have brought to light a number of shortcomings and associated research issues. A number of significant issues flow

from the roots from which PDM systems grew: representation on 'only' a data level for engineering information (Ameri and Dutta, 2005), limited to the management of engineering documents for the design and manufacturing domains. As Bilgic and Rock (1997) note, PDM systems suffer from ambiguous product representation. These systems are adept at representing the engineering bill of materials (EBOM) and manufacturing bill of materials (MBOM), but typically lack the facilities to represent context and usability; in particular, function, behaviour, requirements and geometric representations are buried within the documents that are managed in the electronic vault. As Philpotts (1996) as well as Bilgic and Rock (1997) note, in the middle of the 1990's PDM systems were typically not compliant with the emerging ISO 10303 Standard for the Exchange of Product model data (STEP), so though life cycle functionality was promised, a standard mechanism to achieve lifecycle functionality was lacking. According to Bilgic and Rock (1997), the ambiguity in product representation resulted in more shortcomings of PDM systems: the lack of a possibility to perform analysis on the impact of proposed design changes, a deficiency in representing and classifying function and behaviour (what the components in the product breakdown structure are actually for and how they are used and reasoned upon) and a lack of reuse of design knowledge. Similar points are made by Maropoulos (2003), who notes that "current [PDM] systems offer good capabilities in data management and workflow coordination, whilst they offer very little support in the critical area of knowledge management and representation"; in particular, the management of process knowledge is marked as an area requiring intensive research.

To summarize, PDM faces the following shortcomings:

- Representation limited to the data level; very little support for knowledge management and representation
- Representation lacks context and usability
- Lack of engineering analysis due to proposed design changes, lack of representation and classification of function and behaviour and a lack of reuse of design knowledge

During the late 1990s and early 2000s (Liu and Xu, 2001), the combination of these research challenges and the natural evolution of PDM systems led towards the subsequent development of the field of Product Lifecycle Management, or PLM (Abramovici, 2007; Brandt *et al.*, 2008). PLM aims at defining a holistic, contextualised view of the product and associated processes, and thus takes a step up from the data level towards the information level.

2.2.2 Information through Product Life: Product Lifecycle Management

The concept of Product Life Cycle Management (PLM) appeared in the 1990s with the "aim of moving beyond engineering aspects of a product and providing a

shared platform for the creation, organization and dissemination of product related information across the extended enterprise" (Ameri and Dutta, 2005). It is generally noted that PLM has its roots in PDM and CAD technology (Ameri and Dutta, 2005; Lee *et al.*, 2008), but PLM broadens the scope to include elements of the extended enterprise such as the supply chain, sales and marketing and eventually customers. As such, another root branch of technologies for PLM (Amodio *et al.*, 2008; Brandt *et al.*, 2008; Lee *et al.*, 2008) comes from enterprise management systems (e.g. systems for Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Customer Relationship Management (CRM)).

Based on existing PLM literature (Kiritsis *et al.*, 2003; Ameri and Dutta, 2005; Främling and Rabe, 2005; Thimm *et al.*, 2006; Abramovici, 2007; Jun *et al.*, 2007; Bermell-Garcia and Fan, 2008; Wognum and Trappey, 2008)), the following PLM definition can be aggregated: *an integrated approach using a consistent set of methods, models and IT tools to connect product stakeholders for management of product information, engineering processes and applications over the entire lifecycle of a product, from concept to retirement.* As such, PLM sets itself apart from its progenitors, and in particular PDM, by focusing on the entire lifecycle of a product (Ameri and Dutta, 2005; Abramovici, 2007; Jun *et al.*, 2007), "commencing with market requirements through to disposal and recycling" (Thimm *et al.*, 2006). As a consequence, PLM also aims at connecting various stakeholders over the product lifecycle (Ameri and Dutta, 2005; Bermell-Garcia and Fan, 2008; Lee *et al.*, 2008). PLM promises benefits with regard to product information dissemination and sharing along the lifecycle of a product, including the maintenance and end-of-life phases, where employees and organisations can benefit from more complete, up-to-date information. This is particularly important for industries where products have a long life span, such as the aerospace industry, where it is important to "compile a complete record of maintenance events involving each part, for safety, warranty, especially given multiple owners and upgrades/repairs" (Främling and Rabe, 2005) given the role of these records in airworthiness compliance. Under the guise of 'closed-loop PLM', multiple feedback loops are envisioned from the later stages of a product's life towards designers and producers, who can for instance derive information about modes of use and conditions for retirement (Kiritsis *et al.*, 2003). Figure 2.6 shows the information flows that can occur through product life. Feed-forward relations can be distinguished between beginning-of-life stages, middle-of-life stages and end-of-life stages. To give a practical example for aircraft, product specifications (design stage) are compiled into maintenance manuals (product support stage). There are also feedback information flows, such as product usage and failure information being shared with the Original Equipment Manufacturer (OEM). Often, these feedback flows are indirect in nature – information passes through various stakeholders and platforms before ending up at the OEM.

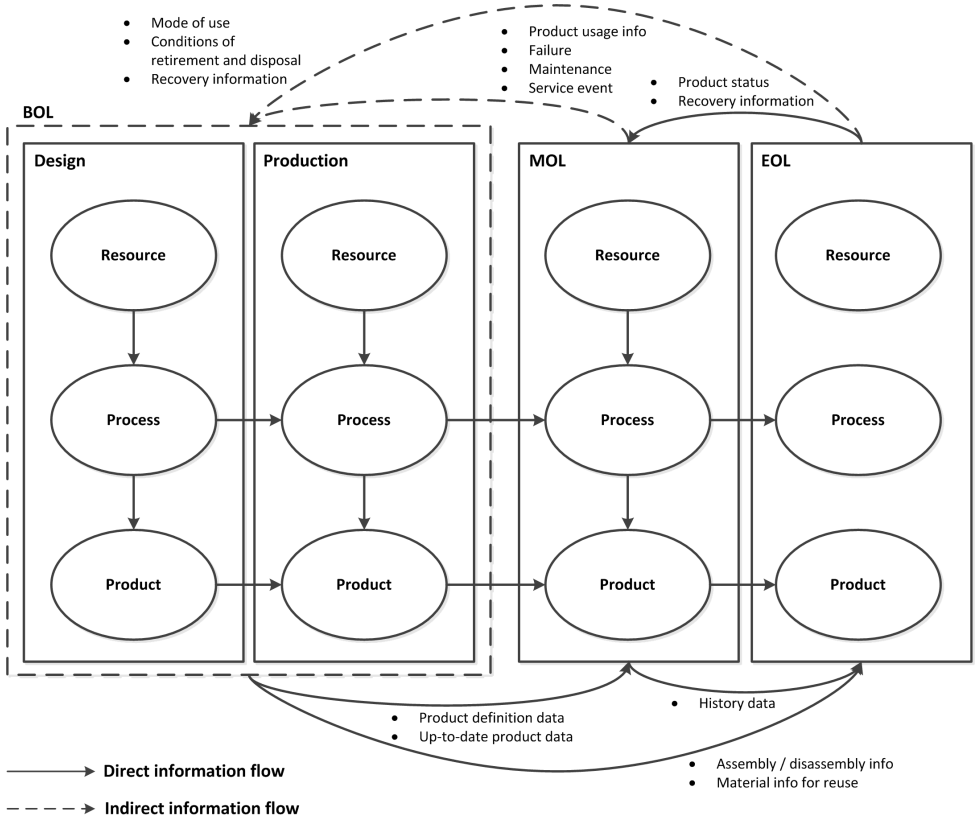

Figure 2.6: Information flows through product life (Jun *et al.*, 2007)

Through the facilitation of information flows between various stages of the product life (e.g. from maintenance to design) , PLM is an enabler for the 'Design for X' approach (Van der Laan, 2008). Other beneficial effects are claimed for PLM, for instance reduced time to market, improved communication among departments and increased success rate of newly introduced products (Lee *et al.*, 2008), but these claims are not substantiated.

Given its wide definition and scope, it is difficult to accurately identify the common functional elements in PLM systems. In practice, PLM often contains a database management system with a centrally controlled data vault in which product information, including models and supporting documentation, can be accessed, managed, maintained and used (Abramovici, 2007; Amodio *et al.*, 2008). Typically, PLM includes an integrated data and process metamodel (Abramovici, 2007) that allows for persistent definition and integrity of product information through product life (Lee and Suh, 2008). It is through this contextualization, as expressed in the integrated data and process metamodel, that PLM moves beyond the data level and towards the information level. To get

there, PLM requires a solid understanding of the semantic (meaningful) and structural characteristics of product lifecycle information, enabling a classification of concepts and relationships which in turn results in a contextual information model or set of models. A critical part of this effort is the identification of the correct product lifecycle metadata for storage during product life. To subsequently use this metadata, methods for the retrieval and analysis of information from lifecycle metadata are in development (Sudarsan *et al.*, 2005; Tomasella *et al.*, 2006; Jun *et al.*, 2007; Matsokis, 2010; Matsokis and Kiritsis, 2010).

Despite progress made over the last decade, PLM still faces significant research challenges. First, contrary to the intended scope of PLM, its adoption is still mainly limited to the product design lifecycle phase, as indicated by Abramovici (2007). This is backed up by Figure 2.7, which shows the use of PDM and PLM throughout the product lifecycle, normalized at 100% for the product design phase. In particular, adoption in the production (18%), delivery (8%) and service (11%) phases is quite low.

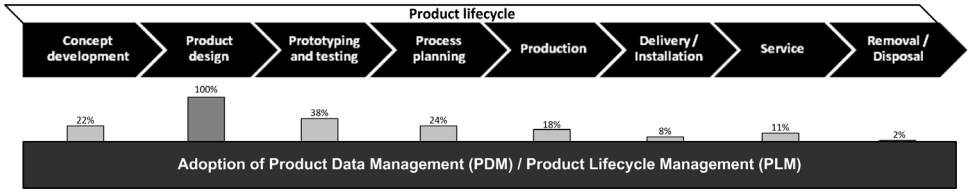

Figure 2.7: Use of PDM and PLM throughout the product lifecycle (Lee *et al.*, 2008)

One can speculate that the limited adoption is due to the PDM and CAD roots of PLM: these technologies originated in and are primarily used in product design, leading to low adoption in the later lifecycle phases. Another factor in the lack of PLM adoption may be a lack of formal modelling techniques and supporting computer languages, as identified by a 2006 survey under PLM experts (Thimm *et al.*, 2006). To alleviate this, research is being undertaken to explore the applicability of different formal modelling techniques and languages as contained in ISO10303 STEP AP 239 - Life Cycle Support (Peak *et al.*, 2004), the Unified Modelling Language (UML) and extensions thereof (Peak *et al.*, 2004; Thimm *et al.*, 2006; Tomasella *et al.*, 2006), as well as ontological modelling (Amodio *et al.*, 2008; Usman *et al.*, 2011), for instance using the Web Ontology Language (OWL)(Matsokis and Kiritsis, 2010). Other authors mention a lack of (model) interoperability (Amodio *et al.*, 2008; Matsokis and Kiritsis, 2010; Ouertani *et al.*, 2011) and a lack of attention to (semantic) structuring of lifecycle information (Jun *et al.*, 2007); a “lack of explicit semantics and context in the information content to be shared across PLM is a major problem” (Ouertani *et al.*, 2011), and “much of the knowledge [sic] is available only in non-structured form” (Amodio *et al.*,

2008). As a result, the traceability of information across different lifecycle phases is limited, where traceability can be defined as 'the ability to describe and follow the life of a conceptual or physical artefact' (Mohan and Ramesh, 2007). Limited traceability leads to limited visibility of product information in middle- and end-of-life lifecycle phases, which makes it more difficult to close the PLM loop from those phases back to the design of new products. Finally, a significant challenge for PLM is to move towards 'smart' systems instead of 'dumb' systems. Currently, information is stored in PLM systems, but the aspect of usability (offering a capability for effective action) is often not addressed.

To summarize, shortcomings with respect to PLM state-of-the-art are:

- Limited adoption beyond the design lifecycle phase
- Lack of formal modelling techniques and supporting computer languages
- Lack of interoperability, explicit semantics and context, and consequently traceability
- Lack of consideration of usability of information

In recent years, research work on PLM has considered the integration of knowledge within PLM systems, for instance as a direct part of PLM systems (as expressed in the European PROMISE project (Bufardi *et al.*, 2005; Främling and Rabe, 2005; Tomasella *et al.*, 2006)), or by integration of Knowledge-Based Engineering (KBE) and PLM (Bermell-Garcia and Fan, 2008). In the following section, the state of the art regarding knowledge (systems) and product life will be considered.

2.2.3 Knowledge through Product Life: Knowledge Management & Knowledge Engineering

The study of knowledge through product life can be positioned in various ways. Here, the characterisation by La Rocca (2012) is adopted, as shown in Figure 2.8. This Figure shows the relative positioning of knowledge management (KM), knowledge engineering (KE) and knowledge-based engineering (KBE) together with associated knowledge technologies, as bullet-listed. KM is shown as the encompassing area, where 'the attention is on the overall goal of nurturing and supporting initiatives that can enable a more efficient and effective use of knowledge assets in the organisation' (La Rocca, 2012). KE is positioned as part of this area, where 'the emphasis is on the acquisition and codification of knowledge' (La Rocca, 2012). KBE focuses on 'the technical development of the KBE application' and can be seen as an extension of Knowledge-Based Systems (KBS) into the engineering design domain (La Rocca, 2012).

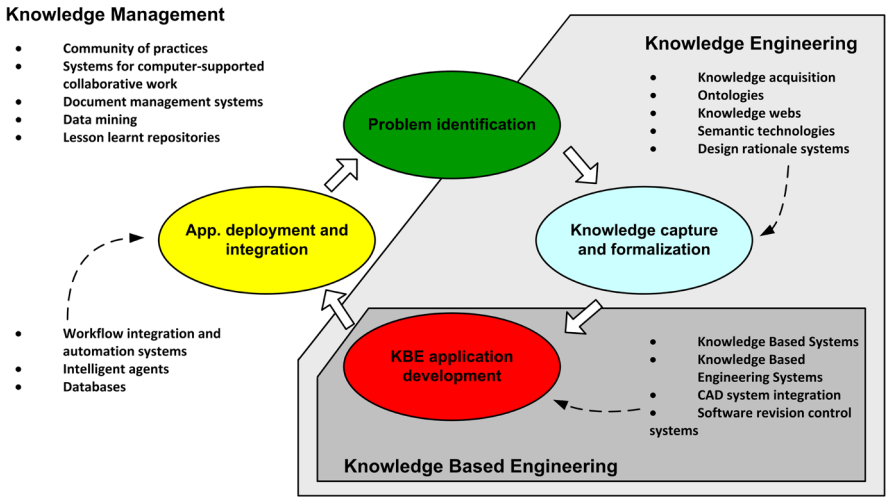

Figure 2.8: Positioning of knowledge disciplines (La Rocca, 2012)

The following discussion follows this positioning. The field of Knowledge Management is discussed first, followed by Knowledge Engineering and Knowledge-Based Engineering.

2.2.3.1 Knowledge Management

From an organizational perspective, handling knowledge throughout product life falls within the scope of Knowledge Management (KM). Literature on KM is varied, but one can distil the following definition for KM: "*a discipline that seeks to improve the performance of individuals and organizations by maintaining and leveraging the present and future value of knowledge assets*" (Newman and Conrad, 2000). The objectives of KM are to make knowledge visible and usable, to develop a 'knowledge-intensive culture' where knowledge is proactively shared, and to build a supporting knowledge infrastructure, including systems and people (Davenport and Prusak, 1998; Alavi and Leidner, 2001). Knowledge management can help organizations in various ways, such as retaining knowledge after loss of key staff and ensuring effective use of structured knowledge, which enables quicker identification, retrieval and leverage of existing company knowledge (Alavi and Leidner, 2001). According to Alavi and Leidner (2001), the basic processes involved in KM are creating, storing/retrieving, transferring and applying knowledge, though alternatives are posited, such as the "Building Blocks for KM" as discussed in Främling and Rabe (2005).

An essential basic KM process is knowledge application, or utilisation, which relates to the "productive effort of organisational knowledge for the organization's sake" (Främling and Rabe, 2005). In order for organizations to derive a capability for effective action – one of the hallmarks of knowledge – it is

vital to know how knowledge behaves and/or changes over time, so that this can be accommodated for in any subsequent knowledge-based application development for deployment in and across organizations, especially for product life cycle-oriented systems. In other words, a proper theoretical understanding of knowledge behaviour through life is required before one can consider management and application of knowledge in knowledge-based applications.

Definitions of knowledge and knowledge change have been discussed at some length in Section 2.1. With respect to the product lifecycle, it is a straightforwardly accepted viewpoint in research that knowledge is created and used during a product lifecycle. This is embodied in the concept of lifecycle knowledge (Ameri and Dutta, 2005) and has been conceptualized into a knowledge lifecycle model based on the technology S-curve (Birkinshaw and Sheehan, 2002), in which knowledge is seen as a resource that is created, mobilized, diffused and subsequently commoditized. In particular, the creation of knowledge has received much attention (see e.g. Nonaka (1994), Nonaka *et al.* (2000), Davenport and Prusak (1998), Alavi and Leidner (2001)), as have the subsequent steps of knowledge capture and formalization (Oldham *et al.*, 1998; Er and Dias, 2000; Stokes, 2001; Preston *et al.*, 2005).

Nevertheless, the aforementioned research efforts do not recognize that knowledge, as a self-contained element, has a lifecycle of its own. Knowledge can be created, used, maintained and retired (Siemieniuch and Sinclair, 2004). Knowledge change throughout product life as experience accumulates has been acknowledged – but no more than that – in knowledge management literature (Geddes and Armstrong, 1991; Nonaka, 1994; Schreiber *et al.*, 1999; Alavi and Leidner, 2001; Simon *et al.*, 2001; Stokes, 2001; Schorlemmer *et al.*, 2002). The same phenomenon is also acknowledged and formally modelled in belief revision literature (Doyle, 1979; Katsuno and Mendelzon, 1991).

However, none of these authors go beyond a qualitative assessment of the differences between static and dynamic knowledge. KM literature lacks a comprehensive model to understand the nature of knowledge and the behaviour of knowledge through life, i.e. knowledge change, is consequently not quantified. Siemieniuch and Sinclair (2004) and Newman and Conrad (2000) discuss some of the possible stages of knowledge. However, they fail to discuss explicit provisions for certain stages of the life cycle of knowledge, for instance formalization, maintenance and retirement, and do not consider transformations between lifecycle stages of knowledge. Theorems for reasoning about knowledge change have been posited as part of literature on belief revision and truth-maintenance systems (Doyle, 1979; de Kleer, 1986; Katsuno and Mendelzon, 1991), but to the best of the author's knowledge the behaviour of knowledge over time has never been quantified using these theorems.

To conclude, one dominant shortcoming in KM literature can be identified: no single consistent model exists to characterise *and* measure knowledge change over its lifecycle, and consequently there are currently no means to quantify the behaviour of knowledge during product life.

2.2.3.2 Knowledge Engineering: KBS & KBE

The discipline of Knowledge Engineering (KE) focuses on the acquisition and codification of knowledge to support the development, implementation and maintenance of Knowledge-Based Systems (KBS) (Studer *et al.*, 1998). KBS are systems that use an acquired and codified set of knowledge to offer problem-solving advice (Expert Systems) or to solve problems directly. KBS are typically comprised of a structured knowledge base containing a body of domain knowledge next to acquisition mechanisms and reasoning mechanisms to solve the problems at hand (Studer *et al.*, 1998). A user interface is provided to allow interaction with users.

Over the past decades, a number of methodologies have been proposed to support the development of KBS, for instance CommonKADS (Common Knowledge Acquisition and Documentation Structuring) (Schreiber *et al.*, 1999), MIKE (Model-based and Incremental Knowledge Engineering), and Protégé-II (Studer *et al.*, 1998; Kuhn, 2010). Perhaps the most well-known of the development methodologies is CommonKADS (Schreiber *et al.*, 1999), a methodology that aims to cover the entire life cycle of knowledge-based systems. CommonKADS comprises a set of models that capture the functional aspects of the KBS as well as the environment in which the KBS will operate (Studer *et al.*, 1998). In particular, the *Knowledge Model* is a core contribution of CommonKADS with respect to the formal modelling of KBS, as it offers a way of structuring and modelling the domain knowledge, inference structure and actions, and task decomposition (see also Chapter 3.3).

Most knowledge-based systems using knowledge engineering methods and techniques are confronted by a number of challenges. From a design engineering perspective, the most notable shortcomings of KBS are that they lack a capability for geometry manipulation and data handling (La Rocca, 2011; La Rocca, 2012). Ideally, the KBS capabilities regarding knowledge capture, knowledge representation and reasoning would be merged with Computer-Aided Design (CAD) and Computer-Aided Analysis (CAA) capabilities to provide engineers with automated assistance in geometry manipulation and data processing. To achieve just this, Knowledge-Based Engineering (KBE) initiatives originated in the early 1980's.

Knowledge-Based Engineering (KBE) can be seen as an extension of KBS into the design engineering domain, adding facilities for geometry manipulation and data handling capabilities (La Rocca, 2012). KBE is characterised by its language-based, object-oriented approach. KBE systems are used as general purpose tools

to develop KBE applications through a programming approach using KBE programming languages. The defining characteristics of these languages are discussed in Cooper and La Rocca (2007) and La Rocca (2012). An important issue to note is that 'KBE applications show no crisp separation between knowledge and inference mechanism' (La Rocca, 2012), meaning that 'expanding, updating and maintaining a KBE application is not just adding or deleting rules from a list' (La Rocca, 2012). Consequently, proper documentation of KBE application code is paramount to avoid a black-box effect (also see below).

The objective of KBE is to reduce time and cost of product development, which is primarily achieved through automation of repetitive design tasks while capturing, retaining and re-using design knowledge (La Rocca and van Tooren, 2009). Table 2.2 summarizes some KBE development efforts as an indication of the potential of KBE.

Table 2.2: Summary of selected KBE development efforts

Source	Subject	Effects
Van der Laan *et al.*, 2005	Parametric Modeling of Movables for Structural Analysis	Up to 80% time savings in FE model generation (from 8 hours to 1 hour for specific instances)
Chapman & Pinfold, 2001	Automotive Body-in-White concept design	BIW mesh generation from 15 man weeks to 'minutes'.
Choi *et al.*, 2005	Composite aerospace structure: cost and weight estimation	Rapid evaluation of cost and weight for composite structures: supports trade-off capability
Kulon *et al.*, 2006	Manufacturing process design: hot forging	New designs in hours rather than days or weeks. Supporting accessible knowledge base.

To identify the research challenges currently faced by KBE, a critical review of existing KBE literature has been performed. For a detailed discussion, please refer to the associated publication (Verhagen *et al.*, 2012). Figure 2.9 summarizes the selection, classification and review process. The selection process has resulted in a consolidated review sample consisting of fifty research contributions. That sample has been reviewed once for the identification and classification of review criteria. These review criteria have been applied during a second review round.

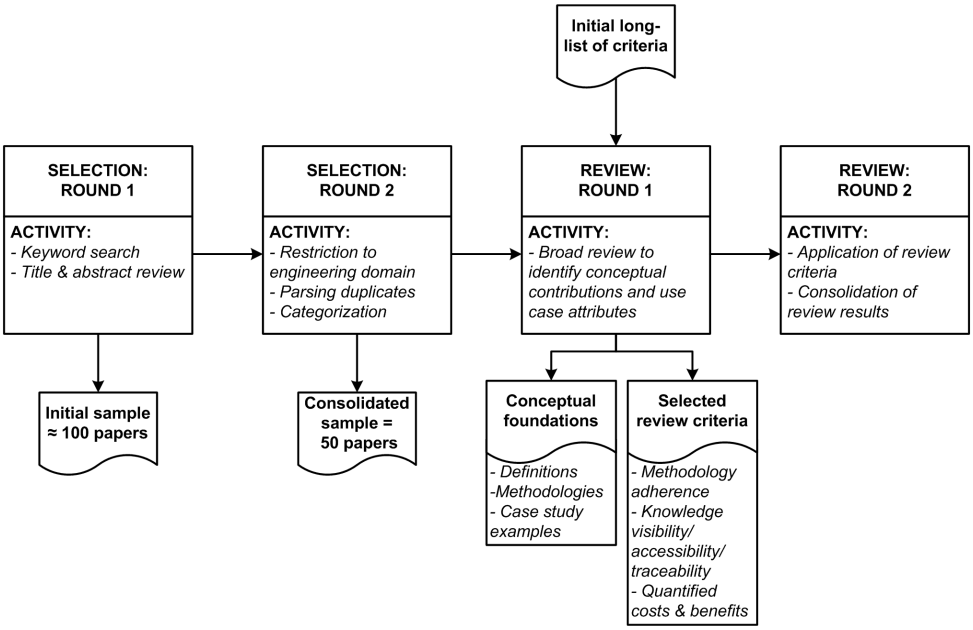

Figure 2.9: Selection, classification and review process

Five major shortcomings of KBE have been distilled from application of the review criteria to the review sample.

1. **Case-based, ad-hoc development of KBE applications:** Development of KBE applications is still very much case based and happens on an ad-hoc basis (Sainter *et al.*, 2000). This is confirmed by the wide-spread non-adherence to KBE design methodologies. From the 37 papers within the sample describing case studies, 81 % (30 papers) did not explicitly adhere to a specific methodology. The practical impact of existing methodologies seems to be limited. The resulting case-based nature of KBE development is a significant problem. It can lead to knowledge loss due to poor modelling of the application and inadequacies in the used development language; it can cause knowledge misuse if the wrong kind of applications are developed; knowledge runs a danger of being under-utilized, due to an inability to share and re-use it at computer and human levels, and finally, maintenance costs will be higher due to non-standard development (Sainter *et al.*, 2000).
2. **A tendency toward development of 'black-box' applications:** Another finding of the review is that current KBE development has a tendency towards 'black-box' applications – many applications (e.g. Choi *et al.* (2005); Kulon *et al.* (2006); Ko *et al.* (2007)) at best represent captured knowledge as context-less data and formulas. There is no explication of

formulas and the actual meaning and context of the captured knowledge, let alone provisions for capturing design intent.

3. **A lack of knowledge re-use:** The previous review findings tie in closely with the difficulty of re-using knowledge in KBE applications. Case-based black-box KBE applications do not particularly invite knowledge re-use. Knowledge re-use is further complicated by the difficulty of sharing knowledge across (KBE) applications and platforms; as Bermell-Garcia and Fan (2008) note, "using current data exchange standards, it is only possible to transfer an instance of the design (one state of the design), and not the knowledge embodied to generate it".
4. **A failure to include a quantitative assessment of KBE costs and benefits:** Most KBE research fails to quantitatively illustrate the advantages and costs of KBE. 25 out of 37 case studies (67%) do not mention the resulting time or cost advantages associated with KBE adoption, let alone the more sensitive information about KBE development cost. An example of a more systematic approach towards KBE quantification has been performed by Corallo *et al.* (2009), who use Activity Based Performance Measurement (ABPM) for cost-benefit assessment of KBE in new product development. Unfortunately, this quantification effort has been performed on a single case study, so validity, reliability and generalizability of the ABPM approach for KBE quantification are not known.
5. **A lack of a (quantitative) framework to identify and justify KBE development:** A final KBE aspect that has not received much attention in literature is the assessment of KBE development opportunities. The MOKA handbook (Stokes, 2001) presents some qualitative criteria for identification and justification of KBE opportunities. Emberey *et al.* (2007) and Van der Elst (2007) use these and more criteria to assess whether a design task is suitable for KBE application development. The Adaptable Methodology for Automated Application Development (AMAAD) of Van der Velden *et al.* (2012) proposes a complexity analysis to ascertain the required level of automation for engineering tasks and associates this level with automation features. However, as the authors themselves note, this approach has significant room for improvement, both for the complexity analysis itself as for the addition of a quantitative analysis aspect (Van der Velden *et al.*, 2012). Despite the mentioned initiatives, no solid framework or method using both qualitative and quantitative aspects is available to determine whether a design task, product or process is suitable to develop a KBE application for. This shortcoming reflects the point made in the previous section regarding the lack of a knowledge life cycle model, and associated quantification.

In the following section, the discussed research issues for data, information and knowledge through product life will be synthesized into a coherent set of research challenges that are subsequently related to the research objectives.

2.3 Identification of Research Challenges

Based on the discussion of related literature presented in Section 2.2, a number of research challenges can be identified from the shortcomings of the current state-of-the-art. These challenges can be subdivided into a theoretical and a practical perspective.

From the theoretical perspective, the central challenge to consider is related to the behaviour of knowledge over time. Is it static or does it change? In other words, does knowledge have a lifecycle of its own? As explained in Section 2.2.3, no existing literature goes beyond a qualitative assessment of the differences between static and dynamic knowledge. A comprehensive model to understand the nature of knowledge through life is lacking, and the behaviour of knowledge is consequently not quantified. To summarize, from the theoretical perspective a first challenge materialises with respect to knowledge:

- **Theoretical challenge:** it is necessary to find out what knowledge precisely means in the context of the product lifecycle and to consequently model and quantify the behaviour of knowledge within product lifecycle stages.

From a practical perspective, knowledge engineering approaches have primarily been adopted in the design phase of the product lifecycle, as embodied in the field of Knowledge-based Engineering (KBE) – see also Section 2.2.3.2. With respect to the product lifecycle, it is possible to discern a practical challenge for knowledge engineering that focuses on the role of knowledge within the development and implementation of knowledge-based applications:

- **Practical challenge: knowledge use and lifecycle management**
  Knowledge, information and data are contained within knowledge-based applications, but these are very often difficult to access, study, directly use and maintain (McMahon *et al.*, 2005). This would be no problem if knowledge is static, but too often knowledge is simply assumed to be static. For most knowledge-based applications, the usability and maintainability of knowledge is not guaranteed, which can lead to rapid obsolescence of these applications as knowledge and functionality become out-dated. As such, the following elements and associated issues

must be addressed to move towards more transparent, useable and life cycle-ready knowledge-based applications:

- **Maintainability of knowledge in knowledge-based applications: moving beyond black-box applications and ensuring transparency:** As mentioned, a current drawback of many KBE applications is that they are 'black-box': the knowledge contained in the KBE applications is difficult to access and inspect, and is often embedded in the application code (Verhagen *et al.*, 2012). To improve it is necessary to move beyond black-box processes and applications by supporting categorization, accessibility and traceability of knowledge, which opens up the potential for knowledge (re-)use (Markus, 2001; Verhagen *et al.*, 2012). The associated necessity for increased transparency in knowledge-based applications in general is a well-noted research issue (Sunnersjo *et al.*, 2006; Fan and Bermell-Garcia, 2008; Elgh and Cederfeldt, 2010). To achieve transparency, knowledge included into knowledge-based applications should be substantiated: the underlying knowledge and supporting documentation for the knowledge-based application should be categorized and be directly accessible. Direct interfacing with knowledge management applications (Fan *et al.*, 2002; Verhagen and Curran, 2011) or PLM solutions (Ma and Liu, 2007; Bermell-Garcia and Fan, 2008; Fan and Bermell-Garcia, 2008) can be used to achieve this, and open up the opportunity to utilize knowledge-based applications from an enterprise context. To enable effective use and update of knowledge, the knowledge element(s) in applications should be formally structured using knowledge model(s) and metadata.
- **Usability of knowledge in knowledge-based applications:** the following issues should be addressed to improve usability of knowledge in knowledge-based applications:
    - **Task orientation:** knowledge involves a capability for effective action. This capability can be met by explicitly associating sets of knowledge with functional tasks. Following from the shortcomings of PDM and PLM, representation of knowledge should allow for context and semantics. Effective action is determined by the completeness of the end result, and the time and resources necessary for achievement of that result. It is here that a knowledge-based approach may produce

benefits by reducing time and resources through automation of repetitive steps.

- **Expert / end user involvement:** assuming that knowledge entities are often not static over time but are subject to change, it is necessary that experts and / or end users – the knowledge owners and "intrinsic components of the knowledge creation and utilization process" (Siemieniuch and Sinclair, 1999) – are actively involved in the management of knowledge embedded in knowledge-based applications. The knowledge owners should not just serve as remote knowledge sources for the knowledge engineers to call upon when necessary. To be actively involved in knowledge management and application maintenance, users should derive a direct benefit from this effort without being burdened with too many tasks. From this perspective, effective and efficient personalization and codification of knowledge must be supported (McMahon *et al.*, 2004): context and semantics of knowledge must be provided to guide users towards knowledge useful to them. End users must be able to identify, use, interact with and if necessary, update the relevant knowledge that they use in their daily workflow and specific context (Merali and Davies, 2001). This requires that a) knowledge is tied to engineering tasks, as noted in the previous point; b) knowledge is visible and directly accessible for end users to enable interaction – context and semantics must be provided.

- **Practical challenge: methodological approach to facilitate knowledge change management**
  Assuming that knowledge change occurs, development of a methodology that takes this change into account is required. Given the previously mentioned lack of methodology adherence in knowledge-based application development, it is a questionable idea to develop a complex, full-fledged new methodology. To defuse this problem, any new methodology must be compatible to existing methodologies and keep complexity to a minimum.

In a nutshell, the research challenges stem from the realization that it is currently impossible to straightforwardly characterize and measure knowledge change. If knowledge does change, from a practical perspective this must be managed –

utilization and maintainability of knowledge take centre stage. A methodological approach to facilitate knowledge change management must be developed.

The presented research challenges can be used in conjunction with the research objectives to present a well-founded rationale for research. In Table 2.3, the research objectives are revisited and related to specific research challenges.

Table 2.3: Research objectives related to research challenges

Research objective	Associated research challenge(s)
Knowledge lifecycle modelling	Characterise, model and quantify the behaviour of knowledge within product life
Ontology-based approach to support knowledge change	Maintainability: - Moving beyond black-box KBS applications and ensuring transparency Usability: - Task orientation - Expert / end user involvement
Methodology development	Methodological approach to facilitate knowledge change management

In the following Chapter, three contributions to theory are developed. These contributions address the introduced research challenges.

3 Theory Development

This chapter is comprised of three parts that address the characterisation, quantification and implementation of knowledge change in knowledge-based applications. In this chapter, the three main research objectives are addressed:

1) ***Knowledge life cycle modelling:*** Section 3.1 addresses the first research objective by focusing on the development of a conceptual model for the lifecycle of knowledge.
2) ***Ontology-based approach to support knowledge change:*** The knowledge lifecycle model is an input for Section 3.2, in which an ontology is developed to facilitate knowledge change (management) in the development and use of knowledge-based applications. This ontology takes into account usability and maintainability aspects.
3) ***Methodology development:*** Finally, Section 3.3 introduces a methodology to guide the development of knowledge-based applications that must cope with knowledge change. In doing so, the third research objective is accounted for.

By addressing these research objectives and the associated research challenges, three main contributions to theory are realized. Table 3.1 relates the research challenges defined at the end of Chapter 2 with the research contributions made in this dissertation.

Table 3.1: Research challenges related to research contributions

Research challenge	Research contribution
Characterise, model and quantify the behaviour of knowledge within product life	Conceptual knowledge lifecycle model (Section 3.1)
Maintainability: - Moving beyond black-box KBS applications and ensuring transparency Usability: - Task orientation - Expert / end user involvement	Ontology-based approach to support knowledge change: the Knowledge Lifecycle ontology (Section 3.2)
Methodological approach to facilitate knowledge change management	KNOMAD methodology (Section 3.3)

The research contributions are discussed in Section 3.4. Subsequently, a case study approach is proposed to validate the research contributions.

3.1 A Conceptual Model for the Lifecycle of Knowledge

This section describes an initial effort at modelling knowledge as a concept with its own lifecycle. It addresses the following questions within the context of the first research objective – knowledge life cycle modelling:

- Which concepts and relationships are required to characterise the change of explicit knowledge within and throughout the aircraft lifecycle phases?

3.1.1 State of the Art and Shortcomings of Knowledge Lifecycle research

In Section 2.3, a challenge regarding knowledge within the context of the product lifecycle has been defined. After application of the research scope, the following challenge can be identified:

- **Theoretical challenge:** it is necessary to find out what knowledge precisely means in the context of the aircraft lifecycle, and how the dynamic behaviour of knowledge within aircraft life can be modelled and quantified.

As noted before, a comprehensive model to understand and quantify the behaviour of knowledge over time has not been developed before. There is not a lot of understanding regarding the 'life' of knowledge: When and how does it change? What are the states and transformations that are involved? To address these questions, relevant insights from literature are combined with original research to come up with a conceptual model of the knowledge life cycle.

Most literature regarding the knowledge lifecycle focuses on the creation, sharing and application of knowledge from an organisational perspective; a knowledge lifecycle is seen as “a methodology or process that produces knowledge” (McElroy, 2003) or as a process of “how organisations generate, maintain and deploy a strategically correct stock of knowledge in order to create value” (Buckowitz and Williams, 1999).

Some examples of these process-oriented (organisational) 'knowledge lifecycle' models are given in Table 3.2. Please note that the model stages are not horizontally equivalent.

Table 3.2: Process-oriented (organisational) knowledge lifecycle models (adapted from Maksimovic *et al.* (2011))

	KLCs in KM		KLCs in KBE	
Models	Buckowitz and Williams (1999)	McElroy (2003)	Stokes (2001)	Rodriguez and Al-Ashaab (2007)
Stages	Get	Individual and group learning	Identify	Identify
	Use	Knowledge claim formulation	Justify	Capture and standardize
	Learn	Information acquisition	Capture	Represent
	Contribute	Knowledge validation	Formalise	Implement
	Assess	Knowledge integration	Package	Use
	Build & Sustain		Analyse	
	Divest			

A significant drawback of these models is that they do not consider knowledge as an independent element that has a life of its own. However, some research efforts describe the conceptual elements of knowledge through life. In particular, Siemieniuch and Sinclair (2004) identify basic states of knowledge: it can be created, used, maintained and retired. Similarly, the General Knowledge Model devised by Newman and Conrad (2000) includes knowledge creation, knowledge retention and knowledge utilisation as key stages.

Knowledge creation relates to “activities associated with the entry of new knowledge into the system, including knowledge development, discovery and capture” (Newman and Conrad, 2000). Knowledge retention covers “all activities that preserve knowledge and allow it to remain in the system once introduced, including maintaining the viability of knowledge within the system”. This point is strongly related to expansion, revision and reduction of belief sets, which is part of belief revision literature (Doyle, 1979; de Kleer, 1986; Martins and Shapiro, 1988). Knowledge utilisation concerns the “activities and events connected with application of knowledge to business processes”.

Most research into knowledge states has focused on the 'Create' state – how is knowledge created by individuals and organisations? In particular, the work of Nonaka (1994) on the SECI model has had a major influence on the research area. Nonaka (1994) maintains that knowledge is created through the conversion

between tacit and explicit knowledge. The SECI model describes the four modes of conversion between tacit and explicit knowledge: socialisation (from tacit to tacit knowledge – creating tacit knowledge through shared experience or interaction between individuals), externalisation (from tacit to explicit knowledge), commoditisation (creating explicit knowledge from explicit knowledge, for instance through sorting, adding, recategorizing or recontextualizing) and internalisation (explicit knowledge into tacit knowledge).

As an additional potential input for a knowledge lifecycle model, the MOKA methodology (Stokes, 2001) includes 'Capture' and 'Formalize' as steps within its KBE life cycle. Both steps are useful in conceptualizing the life of knowledge. The 'Capture' step encapsulates the gathering of domain knowledge – both tacit and explicit knowledge – and structuring it by means of MOKA's ICARE forms (Illustrations, Constraints, Activities, Rules, Entities). The relationships between these various forms of knowledge are modelled. The structured raw information forms the basis of MOKA's Informal Model, which intends to make domain knowledge understandable for both the domain expert and the knowledge engineer. The subsequent step ('Formalize') uses the Informal Model to create a Formal Model: Unified Modelling Language (UML)-based product and process models of the domain knowledge that together are ready for implementation into a knowledge-based application. MOKA is supported by a few knowledge engineering tools, including PC-PACK (Epistemics).

The exploration of knowledge lifecycle concepts can be further supplemented by looking at documentation management literature. This type of literature gives an idea of the progressive actions that can be taken with respect to codified, explicit knowledge. To enable the use, management and maintenance of documents, the possible statuses of a document can be categorized and described. Eynard *et al.* (2004) identify four possible document statuses in the context of an engineering process within a PDM implementation. The four are 'In progress', where data is currently modified and not useable for other activities; 'Shared', where data is deemed sufficiently mature to be used as input for other activities; 'Released', where data is frozen and not further modified; and 'Obsolete', where data cannot be used as input for an activity. A similar but slightly different categorization exists in the work of Gielingh (2005), who identifies the four categories of 'In work' (draft), 'Restricted' (review), 'Final' (use) and 'Revise' (maintenance).

A final addition to the exploration of knowledge lifecycle concepts is found in belief revision literature. Three main kinds of belief kinds can be distinguished: expansion, revision and contraction (Gärdenfors, 2003). Expansion considers the inclusion of a new belief into a belief system. Revision deals with the inclusion of a new belief into a belief system, leading to deletion of some old beliefs in the system to retain consistency. Contraction considers the retraction of an old belief

from a system, which may lead to further retraction to maintain consistency in the belief system.

Summarizing, Table 3.3 gives an overview of potential knowledge lifecycle states that can be used in defining a conceptual model for the knowledge lifecycle. Note that the model stages are not horizontally equivalent. These states will be revisited in Section 3.1.3.

Table 3.3: Potential knowledge states

	Knowledge literature		**Documentation management literature**		**Belief revision literature**
Models	Siemieniuch and Sinclair (2004)	Newman and Conrad (2000)	Eynard *et al.* (2004)	Gielingh (2005)	(Gärdenfors, 2003)
Stages	Create	Creation	In progress	In work (draft)	Expansion
	Use	Retention	Shared	Restricted (review)	Revision
	Maintain	Utilisation	Released	Final (use)	Contraction
	Retire		Obsolete	Revise (maintenance)	

Following upon the definition of knowledge change in Section 2.1, a number of knowledge lifecycle models and stages have been introduced in this section. The organisational lifecycle models do not consider knowledge as an element with a lifecycle of its own. Various authors from knowledge, documentation management and belief revision literature do offer stages to characterise knowledge change.

However, a major shortcoming is that the state of the art does not go beyond a qualitative characterisation of knowledge change. The aim of the next sections is to identify requirements for a knowledge lifecycle model for quantification of knowledge change and subsequently define this model.

3.1.2 Requirements on Definition of a Knowledge Lifecycle Model

In positing a model for the lifecycle of knowledge itself, the shortcomings of previous models should ideally be avoided. Based on the discussion of existing research, the key guideline towards conceptualization of a knowledge lifecycle model is that knowledge should be viewed as an independent element that has a

lifecycle of its own and which must be measured. This is reflected in the following derived requirements for definition of a knowledge lifecycle model:

Types of change (data, information and knowledge change): values associated with knowledge elements must be able to change, as well the structured context (type, number and applicability of relations), plus capability for effective action (identify and measure change in rules, logic or attributes)

1) **The model should reflect the nature of knowledge and knowledge change:** in Section 2.1, knowledge and knowledge change have been defined. To reiterate, the definition of knowledge adopted here is *processed information resulting in a capability for effective action.* The associated types of change (data, information and knowledge change) must be able to be identified and measured.
2) **The model should centre on knowledge as an independent concept:** it should be as free as possible of confounding factors. Organizational processes (e.g. justification of knowledge-based development) or implementation factors (e.g. technical specification of the knowledge base) are important from both academic and practical perspectives, but as the previous Section has shown, research has already performed into these issues. The main focus of the knowledge lifecycle model should be on knowledge itself.
3) **The model should be unambiguous:** the concepts used in the model should be clearly and unambiguously defined. This is a necessary condition to enable understanding and application of the model.

The ultimate objective of the model is to enable the measurement of knowledge change (see Section 2.3 and the introduction of Section 3.1). This is reflected in the following derived requirements:

1) **Be able to operationalize concepts into measures:** to measure something, directly observable measures must either be defined directly in the model, or be definable from the concepts used in the model. In the latter case, the required operationalization must be straightforward and error-proof. This reflects on requirement 3) – the unambiguous definition of model concepts.

2) **Be able to apply it across research domains and lifecycle phases:** the model concepts must ideally be generalizable across research domains and lifecycle phases. This improves the power and applicability of the model. The concepts must be sufficiently abstract to enable wide application.

3.1.3 Research Contribution 1: Conceptual Knowledge Lifecycle Model

To meet the stated requirements, the following concepts are used to construct a Knowledge Lifecycle Model: knowledge states and knowledge actions. States are an 'instantaneous description of an entity' (Umeda *et al.*, 1990) and actions represent one or more (sequential) changes of states. In this, actions are analogous to behaviour as defined by Umeda *et al.* (1990). The use of states and actions can be seen as a potential starting point towards more involved modelling following the Function-Behaviour-State (FBS) framework. For now however, states and actions are sufficient to address the objectives of knowledge lifecycle definition and measurement, as embodied in the requirements posed previously.

1. **Knowledge states:** the concept of knowledge states is a familiar feature in existing knowledge models, as shown in Table 3.3. Knowledge can be in several states along its lifecycle. From start to end of knowledge life, the states conceptualized here are *creation*, *formalization*, *utilization*, *maintenance* or *update*, and *retirement*. In the *creation* state, knowledge is being formed and has not been released for application. In the *formalization* state, the knowledge is codified and made ready for (shared) use. It encapsulates capture, structuring and modelling of knowledge, followed by implementation into a (knowledge-based) application. The formalization state addresses the processing of information inherent in the definition of knowledge. In the *utilization* state, knowledge is actively being used, for instance to solve design problems. The utilization state directly addresses the 'actionability' criterion for knowledge. During the utilization state of knowledge users may realise that the current iteration of the knowledge is insufficient, i.e. previous experience, new insights or additional learning may reveal that knowledge needs to be adjusted. In the *maintenance* state, data and information change are incorporated – values or context associated with knowledge elements may change. In the *update* state, knowledge change as well as data and information change is incorporated: values, context and rules, logic or attributes of a knowledge element may change: the capability for effective action is adjusted. This may also lead to renewed insights: new knowledge is created during the *update* stage. Finally, knowledge may become obsolete, out of date or superfluous. It then

enters its final state – *retirement*. This can for instance mean that knowledge is archived and stored for future reference, but it can also mean that knowledge is straightforwardly discarded.

2. **Knowledge actions:** knowledge actions apply to situations where the element as a whole or its content and/or context is being acted upon. Actions can be used to describe the transitions between knowledge states. The following actions are present during the lifecycle, either as intermediates between consecutive knowledge states or as feedback loops:
    a. **Create:** during the 'create' action, an original idea or synthesis of existing ideas and associated data is formed into a new element of knowledge, with attendant context and content.
    b. **Formalize:** the 'formalize' action represents the codification of tacit and/or explicit knowledge into a formal structure and representation, e.g. in an application.
    c. **Use:** during the 'use' action, the knowledge element is deployed to for instance solve problems. This action can be identified, tracked and measured in knowledge-based applications. Through a check whether a knowledge element has been successfully applied, the effectiveness of knowledge application can be checked. Note that this does not reflect on the validity and reliability of knowledge application and the ensuing results.
    d. **Maintain:** as stated before, a knowledge element holds both context as well as content. As a consequence, two separate actions are identified in the lifecycle of a knowledge element that relate to this observation: 'maintain' and 'update'. The 'maintain' action relates to data and information changes: simple value changes and changes in context (relations) apply. For examples, see Table 2.1. The 'maintain' action moves a knowledge element into the 'Maintenance' state – it is of course possible for an element to again be used after this action.
    e. **Update:** the 'update' action is present when knowledge change applies, possibly in combination with data and information change. Rules, logic or attributes may change, possibly leading to a change in outcomes upon application of the updated knowledge element. The 'update' action moves a knowledge element into the 'Update' state, after which the element can feed back into use.

f. Retire: knowledge is declared invalid or out of date, and is subsequently removed from active use. It is possible for retired knowledge to become active again and be used.

When operationalized, these knowledge actions leave 'fingerprints' during the actual life of a knowledge element. This makes it possible to identify and measure changes in knowledge. For instance, when a knowledge element is maintained in a knowledge base, its descriptors will be modified. Such changes can theoretically be tracked and measured.

The resulting Knowledge Lifecycle Model is shown in Figure 3.1. This figure highlights the knowledge states and actions that map the transitions from one state to another.

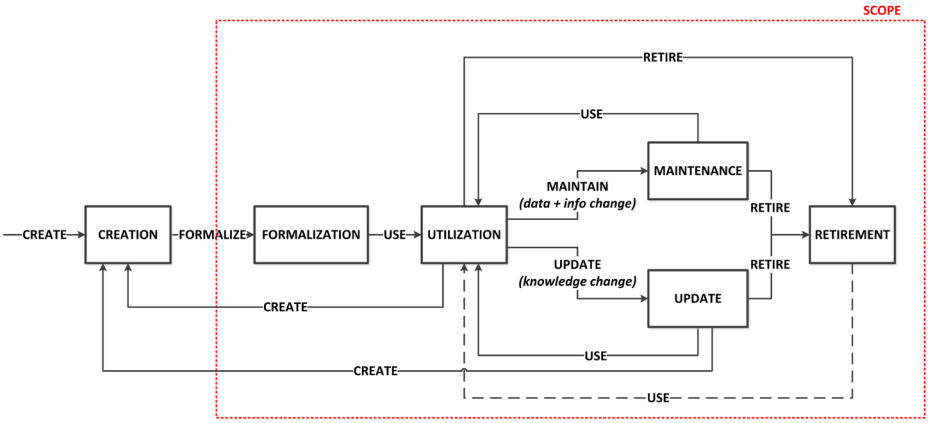

Figure 3.1: Knowledge Lifecycle Model with knowledge states and actions

Given this dissertation's definition of scope (Section 1.2.1.2), the conceptual model will not be tested in its entirety. Instead, the focus is on change in explicit knowledge during the aircraft lifecycle. Within this scope, priorities are to assess identification of knowledge change using the knowledge states and using the concept of knowledge actions to measure knowledge change. With the focus on explicit knowledge, the 'Creation' stage is left out of consideration in the dissertation. The result of this application of scope onto the Knowledge Lifecycle Model is shown through the bounded area in Figure 3.1.

3.1.4 Concluding Remarks

The requirements on the knowledge lifecycle model are addressed by knowledge states and actions in the following manner:

1) **The model should reflect the nature of knowledge and knowledge change:**

The knowledge states *Update* and *Maintain* as well as the associated *update* and *maintain* actions explicitly acknowledge and are able to represent knowledge change.

2) **The model should centre on knowledge as an independent concept:** the knowledge lifecycle model focuses on knowledge as the central conceptual element, with states and actions to describe its various representations.
3) **The model should be unambiguous:** most of the knowledge states and actions are straightforward. A formal (mathematical) foundation for the model is to be developed.

Knowledge actions enable the measurement of knowledge change, as addressed with respect to the following requirements:

1) **Be able to operationalize concepts into measures:** knowledge actions can fairly easily be operationalised into measures, as evidenced in Section 6.2.2.
2) **Be able to apply it across research domains and lifecycle phases:** The state and action concepts are sufficiently abstract to enable application in various aircraft lifecycle phases. Evidence for this is given in Chapters 4-6, where knowledge change is discussed for different lifecycle phases.

The Knowledge Lifecycle Model uses the knowledge state and action concepts and relationships to characterise the change of explicit knowledge. This partly answers the research question as given in the introduction of Section 3.1. However, further questions remain regarding the change of knowledge within specific aircraft lifecycle phases and regarding the quantification of knowledge change. This is discussed in more detail in Section 3.4.1.

3.2 A Model-Based Approach to Support Knowledge Change: the Knowledge Lifecycle Ontology

The Knowledge Lifecycle Model provides a means to characterise and quantify knowledge change. However, this is not enough. If knowledge does indeed change, what can be done to accommodate for this in knowledge-based applications?

This Section addresses the second research objective expressed in Section 1.2 by describing the development of a model-based approach for the support of knowledge change within knowledge-based applications for specific aircraft lifecycle phases. First, challenges related to knowledge-based application development are associated with requirements on the model-based approach. This approach must support the consistent formalization, use and maintenance of changing aircraft knowledge in its various lifecycles. To achieve this, the *ontology* concept is selected as the primary means to express the approach, as discussed in Section 3.2.1. Following this, three main elements are considered as inputs for the design of the ontology. These elements are synthesized in Section 3.2.3, where the ontology is designed and implemented.

The following research questions are addressed throughout this section:

- Which concepts support the consistent formalization, use and maintenance of changing knowledge throughout the aircraft lifecycle?
- How can knowledge change be accommodated during knowledge-based application development?
    - Which models are required and how do these models help to accommodate knowledge change?

3.2.1 The Role of Ontologies in Supporting Knowledge-Based Applications through Knowledge Life: State of the Art and Shortcomings

When assuming that knowledge is dynamic both in and over the aircraft lifecycle phases, a primary consequence is that knowledge-based applications must be able to cope with changing knowledge in and over multiple lifecycle phases. As mentioned in Section 2.3, practical challenges regarding usability and maintainability result from this. Applications must have a

1) Capability to represent a usable function (*usability*)

2) Capability to annotate a function in order to be able to find, access, inspect and maintain this function (*usability & maintainability*)

The maintainability and usability challenges are converted into requirements for the model-based approach. These are derived from the discussion in Section 2.3 and from various literature sources. Table 3.4 presents the conversion from challenges to requirements, with references where appropriate.

Table 3.4: Challenges and associated requirements on the model-based approach

Challenge	Associated requirements
Maintainability of knowledge in knowledge-based applications: **Moving beyond black-box applications and ensuring transparency**	**1) Traceability (Ouertani *et al.*, 2011)** i. **Visibility:** experts / end users should have the possibility to see knowledge that feeds into KBS. ii. **Accessibility:** experts / end users should be able to access and maintain knowledge. iii. **Retrievability:** experts / end users should be able to effectively search for and find relevant knowledge (given that up to 70% of engineers' time is spent searching for knowledge (Lee and Suh, 2008). **2) Functionality (Bermell-Garcia *et al.*, 2012):** experts / end users should be able to inspect and verify the objectives and functions that a knowledge-base application addresses. The required knowledge and processes for achieving functionality must be retrievable and verifiable.
Usability of changing knowledge in knowledge based applications: **1) Task orientation**	**1) Separation of task and domain knowledge (Mizoguchi *et al.*, 1995)**: making domain knowledge independent from an engineering task offers the ability to maintain, update and reuse the knowledge. In other words, a modular approach may be beneficial. It must be stressed that even when task and knowledge are separated, they can still be explicitly associated with each other through modelling of relations. For KBE,

	a modular approach separating task and domain knowledge has been researched by La Rocca (2011) through the concepts of High-Level Primitives and Capability Modules.
2) Expert / end user involvement	**2) Knowledge management** i. Across **domains:** users and tools from different domains should work on the basis of consistent data, information and knowledge – a unified model must be developed. Several research initiatives have made progress towards a unified model for the product lifecycle, for instance the PROMISE consortium (Kiritsis *et al.*, 2003; Tomasella *et al.*, 2006). In the aerospace domain, DLR has developed the Common Parametric Aircraft Configuration Schema (CPACS), which is a data definition schema for describing the characteristics of aircraft, rotorcraft, engines, climate impact, fleets and mission (Rizzi *et al.*, 2012). The iPROD project (iProd, 2013) aims to integrate management of product heterogeneous data. In summary, a sufficiently generalized model that can be consistently applied in multiple lifecycle phases makes it possible to work on the basis of consistent representation of data, information and knowledge. ii. Across **users / actors** from different perspectives (Gielingh, 2005): experts from different lifecycle phases may have different views and priorities when regarding a single reality. Gielingh (2005) maintains that knowledge objects must be recognizable throughout life.

There are some common elements throughout the requirements for the model-based approach. Each of them – traceability (incorporating visibility, accessibility, retrievability), functionality, separation of tasks and domain knowledge, and knowledge management across domains and users – requires that knowledge is structured and made explicit, is kept consistent within and throughout the various aircraft lifecycle phases, and is represented such that users from different domains can find, use and add to their specific knowledge. To meet these requirements, an *ontology* will be used to realize the model-based approach.

An ontology can be defined as 'a definition of a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and the relations among them' (Noy and McGuinness, 2009). Key elements in this definition are the common vocabulary and basic concepts and relations. An ontology necessarily includes a common vocabulary of terms and a specification of their meaning (Uschold, 1996). Without specification, the set of ontology concepts would be variously interpretable by different sets of users. With specification, different users (e.g. experts in different lifecycle phases) with different views on a single reality can be accommodated.

A similar but more complete definition has been proposed by Gruber (1993): ontologies are 'explicit (formal) specifications of a conceptualization'. The 'specification' element has already been discussed. The other three elements in this definition need further clarification. As Uschold (1996) notes, a conceptualization can be seen as 'a world view, a way of thinking about a domain that is typically conceived and/or expressed as a set of concepts, their definitions and their inter-relationships'. An ontology is explicit when it is or can be articulated, coded and stored in certain media, and readily transmitted to others. Finally, the formality of the ontology indicates the level of expression in an artificial, formally defined language, which extends to the possible ontology property of being machine-interpretable. This property is desirable from a functional viewpoint, as being machine-interpretable offers potential for re-use and automation of functionality. Ontologies can be expressed along a range of formality degrees; this is one of the three key dimensions along which ontologies vary, as mentioned by Uschold (1996):

- **Formality:** the degree of formality by which a vocabulary is created and meaning is specified. Uschold (1996) posits a formality continuum that moves from highly informal (loose expressions in natural language) via structured informal (restricted and structured form of natural language) and semi-formal (expressed in an artificial formally defined language) to

rigorously formal (meticulous definition of terms with formal semantics, theorems and proofs of properties such as soundness and completeness).

- **Purpose:** Uschold and Gruninger (1996) identify three main categories of use for ontologies: communication, interoperability and achieving system engineering benefits.
- **Subject matter:** Uschold (1996) identifies three main categories, namely domain ontologies, task/problem solving ontologies, and meta-ontologies. The latter are also called foundational ontologies (Borgo and Leitão, 2007).

To develop an ontology, a number of ontology construction methodologies are available. Examples include the methodologies by Uschold (1996), Noy and McGuinness (2009), Uschold and Gruninger (1996) and the METHONTOLOGY methodology (Fernandes Lopez *et al.*, 1997). All these methodologies share common steps, though the exact representations may vary from methodology to methodology. The common steps have been summarized by Pinto and Martins (2004):

1. **Specification:** identification of the purpose and scope of the ontology.
2. **Conceptualization:** identification of the domain concepts and the relationships between concepts.
3. **Formalization:** organizing the concepts into class hierarchies and subsequent construction of axioms to formally model the relationships between concepts.
4. **Implementation:** codification of the class hierarchies and axioms into a formal knowledge representation language.
5. **Maintenance:** updating and correcting the implemented ontology.

A number of languages have been developed specifically to express ontologies, such as the Knowledge Interchange Format (KIF) (Cranefield and Purvis, 1999) and the Web Ontology Language (OWL). The latter is an example of a language where description logic (DL) can be employed to express predicates that define a concept or a relationship. These predicates allow for inferences to be made on the knowledge base by automatic reasoning. Cranefield and Purvis (1999) give examples of such inferences, including subsumption (generality of concepts relative to each other), coherence (logical coherence of a concept), identity (checking whether two or more concepts are actually expressing the same concept) and compatibility (checking whether two concepts can have common instances). These reasoning capabilities are potentially powerful when constructing an ontology: the integrity and consistency of the ontology can be checked automatically and the use of DL predicates can be used to integrate

different views on a subject through the equivalency of classes (Matsokis and Kiritsis, 2010). Furthermore, querying of an ontology can be assisted by these inferences. In short, languages such as OWL are very suitable for the construction of formal ontologies.

However, a drawback of these specific ontology languages is that they 'are usually defined in terms of an abstract (text-based) syntax and most care is spent on the formal semantics' (Brockmans *et al.*, 2004). There are only limited means to visually model ontologies. Furthermore, the languages and associated tools such as Ontolingua and Protégé are reportedly not very well known outside of the Artificial Intelligence and Knowledge Engineering communities (Cranefield and Purvis, 1999).

In light of these disadvantages, the Object Management Group's Unified Modelling Language (UML) (Object Management Group) is increasingly used to express ontologies. UML is a general-purpose modelling language for object-oriented software-intensive systems. UML can be used to model the static view (structure) of a system (objects, attributes, operations, relationships) as well as the dynamic view (behaviour), which encompasses collaborations between objects as well as changes to the internal state of objects. There are some fundamental and some subtle differences between UML and ontology languages such as OWL. However, distinct advantages of UML with respect to specific ontology languages are:

1) UML is widely known and used throughout industry and academia;
2) UML has a standard graphical notation;
3) UML (and adapted variants) are used in the CommonKADS and MOKA methodologies (see Section 3.3.1); knowledge engineers are familiar with the language and it has proven to be effective in modelling and developing knowledge-based systems.

A significant disadvantage of UML is that it does not support reasoning on its own. To remedy this, an ontology expressed in UML can be extended with predicates using the Object Constraint Language (OCL), which therefore allows for the definition of formal ontologies. However, the UML-OCL combination has some drawbacks in terms of reasoning capability, such as computational complexity.

When properly developed and implemented, ontologies can serve as the backbone for knowledge-based applications. They offer the possibility to structure the knowledge base by modelling the context in which knowledge is viewed. Domain (meta)models are made explicit and knowledge (re-)use is made possible (Matsokis and Kiritsis, 2010). Furthermore, as mentioned, ontologies can incorporate the use of predicates and an inference capability, which offers the potential to execute automated reasoning upon the knowledge base. Finally, ontologies are flexible and can be extended (Brandt *et al.*, 2008). As such,

ontologies not only support multiple viewpoints on the same knowledge, but also offer critical functionality for knowledge-based applications.

Therefore, the model-based approach will be realized through the development of an ontology: the Knowledge Life Cycle (KLC) ontology. Based on the requirements as discussed in Table 3.4, the following associated ontology requirements can be identified (see Table 3.5).

Table 3.5: Ontology requirements

General requirements	Associated ontology requirements
1. Traceability **i. Visibility:** experts / end users should have the possibility to see knowledge that feeds into KBS. **ii. Accessibility:** experts / end users should be able to access and work with knowledge. **iii. Retrievability:** experts / end users should be able to effectively search for and find relevant knowledge.	• Development of metamodel: Explicit representation of semantic (i.e., meaningful) context allows for identification and 'smart' search of information and knowledge
2. Functionality & Separation of tasks and domain knowledge	• Separation into functions and associated task class(es) and extension possibilities for domain ontologies
3. Knowledge management **i.** Across **domains:** users and tools from different domains should work on the basis of consistent data, information and knowledge. **ii.** Across **users / actors** from different perspectives: the model should accommodate the perspectives from different users (e.g. production engineer and machine operator in the manufacturing domain), for the different lifecycle stages.	• Provide common, unified and consistent understanding of knowledge structure. • Accommodate multiple views

To meet these requirements, a number of possible building blocks for the KLC ontology are inspected in the following section. The common steps of the ontology development methodologies (Pinto and Martins, 2004) will be applied together with these building blocks in Section 3.2.3 to develop the KLC ontology.

3.2.2 Main Elements for the Development of the Knowledge Life Cycle Ontology

In this Section, the main building blocks for the development of the ontology are discussed. In the first part, existing lifecycle paradigms and ontologies are discussed. These models provide inspiration for meeting the maintainability requirements associated with the KLC ontology, in particular with respect to the development of a metamodel for traceability and knowledge management. In the next subsection, a number of models are introduced that can be used to address the usability requirements, mainly through the consideration of functionality. Finally, the Knowledge Lifecycle Model from Section 3.1 is introduced as a building block for the KLC ontology. Inputs from the various models are combined in the design and implementation of the KLC ontology (Section 3.2.3).

3.2.2.1 Existing Lifecycle Ontologies[‡]

Most of the recent work into lifecycle models is performed as part of Product Lifecycle Management (PLM) research. For instance, Lee and Suh (2008) propose an ontology-based multi-layered knowledge framework for PLM. Their work first describes previous research regarding this subject, where it is noted that the proposed knowledge frameworks do not include explicit semantics (e.g. Xue *et al.* (1999); Roy *et al.* (2001); Sudarsan *et al.* (2005)) or do not consider the full lifecycle (e.g. Borst *et al.* (1997); Kitamura *et al.* (2004)). Similarly, the multi-layered knowledge framework of Lee and Suh (2008) covers the design and manufacturing domains, but is not extended to the product support and disposal domains. In a similar vein, the ONTO-PDM (Product-driven ONTOlogy for Product Data Management) effort by Panetto *et al.* (2012) proposes an ontological model of a product based on ISO and IEC standards to facilitate the interoperation of application software that share information during the physical product lifecycle, but this model is based in a manufacturing environment and not designed and employed to take into account a lifecycle perspective.

More complete lifecycle modeling efforts have been produced by the PROMISE consortium (short for PROduct lifecycle Management and Information tracking using Smart Embedded systems). PROMISE is a European Framework Six (FP6) research project conducted from 2004-2008. The results of PROMISE (Bufardi *et al.*, 2005; Främling and Rabe, 2005; Tomasella *et al.*, 2006) are considered state-of-the-art in the context of the current research.

PROMISE focused on researching *closed-loop* PLM: a perspective on PLM where information of the whole product lifecycle is tracked and managed. Flows of information can feed forward and backward between different lifecycle stages

[‡] Note: In this Section and consecutive section, ontology classes are expressed in the following format: **Nameofclass, Nameofclass_Addition**. When other conventions have been used in sources, the original formats are translated to the **bold c**lass format.

(BOL, MOL, EOL). As part of the initial results of PROMISE, Jun *et al.* (2007) have summarized research issues on closed-loop PLM per lifecycle phase. This has informed the research direction of PROMISE: high focus was given on the development of a *Semantic Object Model* (Tomasella *et al.*, 2006) or SOM. The SOM takes inspiration from a number of pre-existing standards, including ISO10303 – STEP (Standard for the Exchange of Product model data), ISO 14649 – STEP NC (STEP Numerical Control), ISO 10303-239:2005 – PLCS (Product Life Cycle Support), ISO 15531 – MANDATE (MANufacturing DATa Exchange) and PLM XML, amongst others. The SOM is object-oriented and expressed in UML. The UML SOM model has been converted into a OWL-DL (Web Ontology Language – Description Logic) ontology, with the benefit of adding reasoning capabilities (Matsokis and Kiritsis, 2010).

The SOM is shown in Figure 3.2. Here, one can distinguish two main areas of interest (Tomasella *et al.*, 2006): the area bounded by the continuous line comprising information on product instances and product type, and the area bounded by the dotted line comprising information connected to the different lifecycle phases. The PROMISE SOM is particularly interesting from the viewpoint of the KLC ontology as it contains a high number of relevant classes, properties and relationships that can be used in a metamodel, in particular the **Physical_Product**, **Life_Cycle_Phase**, **Resource (Document_Resource, Personnel_Resource, Equipment_Resource, Material_Resource)**, **Event** and **Activity** classes and their properties and relationships. Furthermore, industry and academia are familiar with its contents, making (part of) its content suitable as a building block for the KLC ontology.

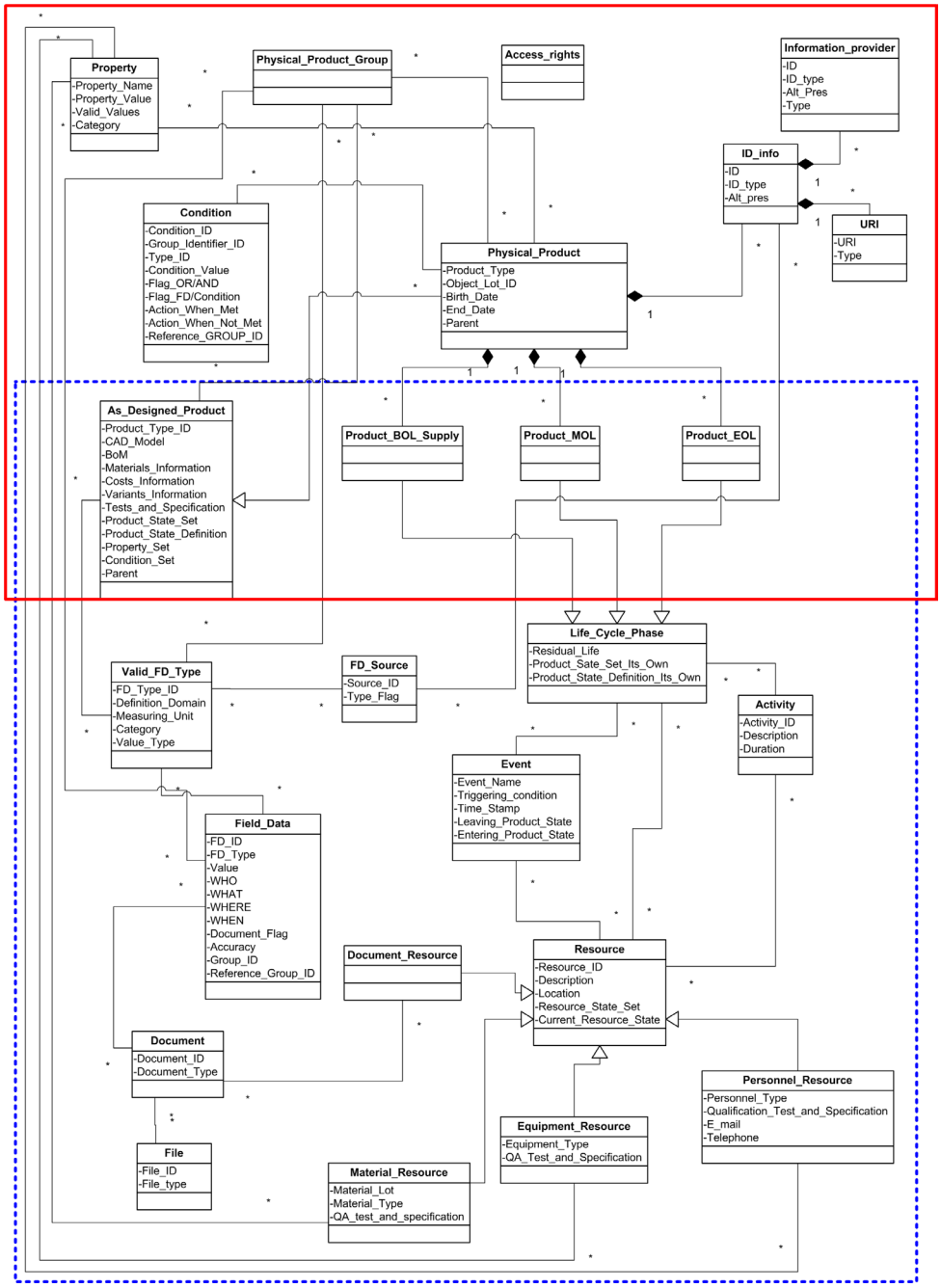

Figure 3.2: PROMISE Semantic Object Model (Tomasella *et al.*, 2006)

Another model with some similarities to PROMISE SOM is the Core Ontology for Process Data Warehouse (Brandt *et al.*, 2008). This model supports "creative,

non-deterministic design processes" by including semantic mechanisms to retrieve and represent knowledge (content), as well as providing "capture and archival of work processes" (Brandt *et al.*, 2008) (i.e. process rationale). The Core Ontology for PDW is given in Figure 3.3. The ontology's central aspect is an **Object** class which can be extended to represent four main areas: process, product, description and storage. Within these areas, similar classes as in the PROMISE SOM appear, for example **Activity**, **Product**, **Document,** and **User**. These and other classes can be extended using ontology modules, represented by the yellow 'folder' representations in the figure (e.g., a document management ontology module can extend the **Store** class).

A novel addition of the Core Ontology with respect to PROMISE SOM is the inclusion of an explicit **Process_Trace** class, which expresses the design rationale by "describing the concrete actions performed in a project by **User** or **Tool**" (Brandt *et al.*, 2008). Furthermore, the Core Ontology makes the storage of documents explicit by including a **Store** class. Both are useful potential additions to the PROMISE SOM as well as potential contributions to the KLC ontology.

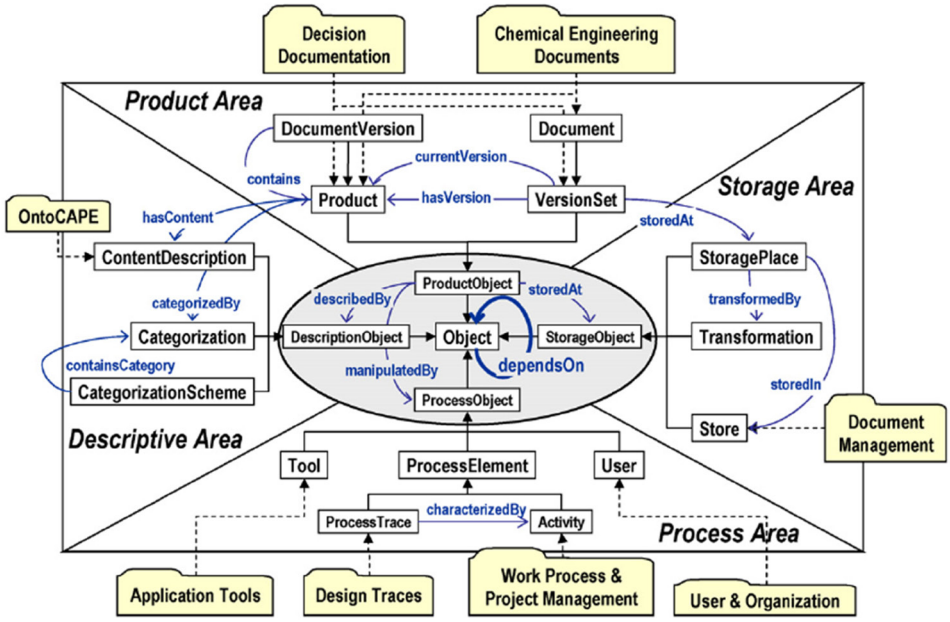

Figure 3.3: PDW Core Ontology (Brandt *et al.*, 2008)

Finally, when considering commercial models, various engineering applications developed by Dassault Systèmes (e.g. Catia™ (Dassault Systemes, 2012) and Delmia™) use the *Product – Process – Resource* (PPR) model. As Butterfield *et al.* (2012) note, this model separates product development into the

three domains of Product, Process and Resource (PPR), enabling the construction of object-oriented tree structures capable of modeling the hierarchies of and all logical relationships between the process, product and resource data (Curran *et al.*, 2010). Through the creation of "links between nodes embedded in the PPR structure, e.g. linking a component in Product to a method in Process and a tool in Resource, it becomes possible to generate active, and importantly interrogable, definitions and quantifiable parameters for design decisions within a virtual environment" (Butterfield *et al.*, 2012).

Together, the PPR paradigm, PROMISE SOM and PDW Core Ontology offer a range of classes and relationships that can serve to fulfil the KLC ontology requirement of having a semantic metamodel. This metamodel can fulfil the traceability and knowledge management requirements, as it can be used to annotate engineering functions or tasks and make them traceable, retrievable, accessible and subject to change for users from different domains.

3.2.2.2 Existing Models for the Representation of Functionality

In this section, existing models for the representation of functions and/or tasks are inspected. Before doing this, it is necessary to define 'function' and 'task'. Interestingly, as noted by Erden *et al.* (2008) in their review of functional modelling, Umeda *et al.* (1995) state that "there is no clear and uniform definition of a function, and moreover, it seems impossible to describe function objectively". In defining the Function-Behavior-State model, Umeda et al. (1990; 1995; 1996) have attempted to define function including subjectivity. Other models have been developed (as reviewed by Erden *et al.* (2008)) that also include subjectivity into function. In these views, function is considered as "a subjective category that links the human intentions/purposes residing in the subjective realm to the behaviours and structures in the objective realm" (Erden *et al.*, 2008). Other definitions of function do not incorporate subjectivity; for instance, function can also be defined as "a relationship between input and output of energy, material, and information" or as "to do something, a combination of verb and noun" (Erden *et al.*, 2008).

The latter definitions overlap somewhat with the concept of tasks as defined by Wood (1986). Tasks can be defined starting from at least four theoretical frameworks: task qua task, task as behaviour requirements, task as behaviour description and task as ability requirements. Wood (1986) selects an approach from these frameworks and arrives at the postulate that all tasks contain three essential components: *products*, *acts*, and *information cues.* Products are "entities created or produced by behaviors which can be observed and described independently of the behaviors or acts that produce them" (Wood, 1986); they are the outputs associated with a task. Acts are either a specific activity or a "complex pattern of behaviour with an identifiable purpose" (Wood, 1986). Wood (1986) considers acts to be an input to a task, but acts are similar to the transformation between input and output mentioned in Erden *et al.* (2008) and

also conceptually close to the Function-Behaviour relationship introduced by Umeda *et al.* (1990). Finally, information cues are pieces of information about object (attributes) that can be used by individuals in the judgements required for carrying out a task. As such, information cues are inputs for a task. Schreiber *et al.* (1999) posit a different definition for a task: in their view, a task is "a subpart of a business process that represents a goal-oriented activity adding value to the organization; handles inputs and delivers desired outputs in a structured and controlled way; consumes resources; requires knowledge and other competences; is carried out according to given quality and performance criteria; and is performed by responsible and accountable agents".

No uniform and universally accepted definitions of function and task relative to modelling exist; views on the two can be quite similar. To make matters more confusing, the terms task, activity and process are often used interchangeably. A more involved discussion of functions and tasks can be found in Erden *et al.* (2008) and Wood (1986). Given the ambiguity regarding 'definite definitions' for these concepts, prescriptive definitions for function and task are avoided. However, to register as either function or task, a concept must include the central aspects of input, activity / process, output and goal.

With the preceding discussion in mind, the first representation of functions and tasks discussed here is the IDEF0 modelling method. The IDEF0 modelling method was adopted as a standard in 1993 by the National Institute of Standards & Technology (NIST) in 1993 (National Institute of Standards and Technology, 1993). Even though it has been withdrawn in 2008 as a federal US standard, it is still commonly applied in academia and industry. The objectives of IDEF0 are to provide "a means for completely and consistently modelling the functions (activities, actions, processes, operations) required by a system or enterprise, and the functional relationships and data (information or objects) that support the integration of those functions" and to provide "a modelling technique which is independent of Computer-Aided Software Engineering (CASE) methods or tools, but which can be used in conjunction with those methods or tools". IDEF0 uses function boxes and inputs, controls, outputs and mechanisms (ICOM) arrows for the representation and modelling of functions – see Figure 3.4 for a generic IDEF0 diagram. In IDEF0 diagrams, functions are defined as "an activity, process, or transformation identified by a verb or verb phrase that describes what must be accomplished". Input are "the data or objects that are transformed by the function into output", Control are "conditions required to produce correct output", such as directions or constraints, Output are "the data or objects produced by a function" and Mechanism are "the means used to perform a function", such as people or machines (National Institute of Standards and Technology, 1993).

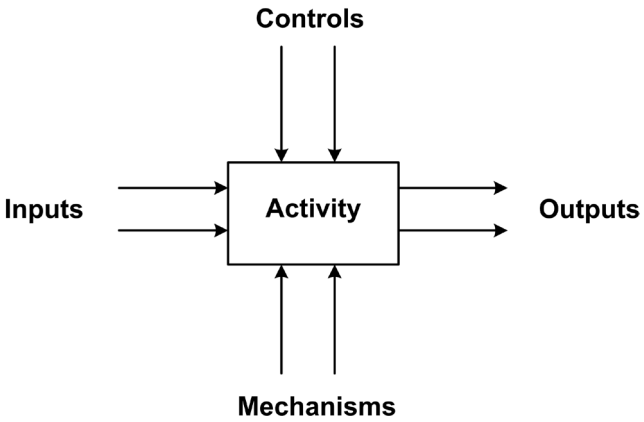

Figure 3.4: Generic IDEF0 diagram (National Institute of Standards and Technology, 1993)

IDEF0 can be used to identify and model functions. Activities can concisely be described by their inputs, outputs, controls, and mechanisms, or ICOMs (Integrated Definition Methods, 2012). Furthermore, IDEF0 can be used to produce hierarchical and increasingly detailed models of functions, activities and tasks. Drawbacks of IDEF0 are that the resulting models can become so concise that non-experts cannot understand them, and that IDEF0 models are commonly interpreted as a sequence of activities, whereas the time dimension is actually not modelled within IDEF0.

The second representation of tasks can be found within the CommonKADS methodology for knowledge engineering (Schreiber *et al.*, 1999). This methodology contains an approach to knowledge modelling that recognizes task knowledge and inference knowledge as specific knowledge categories. Task knowledge is used to describe what goal(s) an application pursues, and how these goals can be realized through decomposition into subtasks and inferences. The inference knowledge represents the lowest level of functional decomposition; it describes the inference steps that are applied to domain knowledge to achieve a reasoning capability.

The General Technology Ontology (GTO), developed by Milton (2007) as an extension to MOKA, can similarly be used to represent tasks. Besides the concept of tasks, the ontology holds 16 other concepts (including resource concepts such as people, software, information, and process concepts such as events, triggers and decision points) that together provide a generic ontology for the construction of knowledge bases. The GTO has been used in the PEGASUS project (PEGASUS, 2013).

An ontological representation of tasks is currently being developed in the iPROD research project (Chan, 2013; iProd, 2013). Here, task ontologies are developed for general, high-level activities: for instance, the design task is

associated with concepts such as requirements, processing and optimization, amongst others. Besides task ontologies, problem-specific domain ontologies are built as a supplement to the task ontology to provide a sufficiently modelled overview of the domain to base subsequent development on. The problem-specific ontologies can be mapped into the higher-level task ontologies.

The final representation of interest for the KLC ontology is formed by the concept of an *Enterprise Knowledge Resource*. This concept has its roots in the work by Bermell-Garcia (2007), who has advocated the annotation of KBE code, applications and/or models (or eXecutable Knowledge Models – XKMs) with metadata, enabling knowledge to be indexed and retrieved in data repositories, PDM and PLM systems. The resulting annotated knowledge models are known as *Enterprise Knowledge Resources* or EKR: a 'specialized type of data resource that automates engineering tasks'. In Bermell-Garcia (2007)'s work, an EKR was represented by "a file containing the code of the XKM and an instance of the metamodel [annotation metamodel] as its blueprint within and [sic] enterprise repository". The proposed metamodel consists of a structure metamodel and an operation metamodel. The structure metamodel can be used to describe the composition of a KBE resource. The operation metamodel can be used to describe how the KBE resource operates.

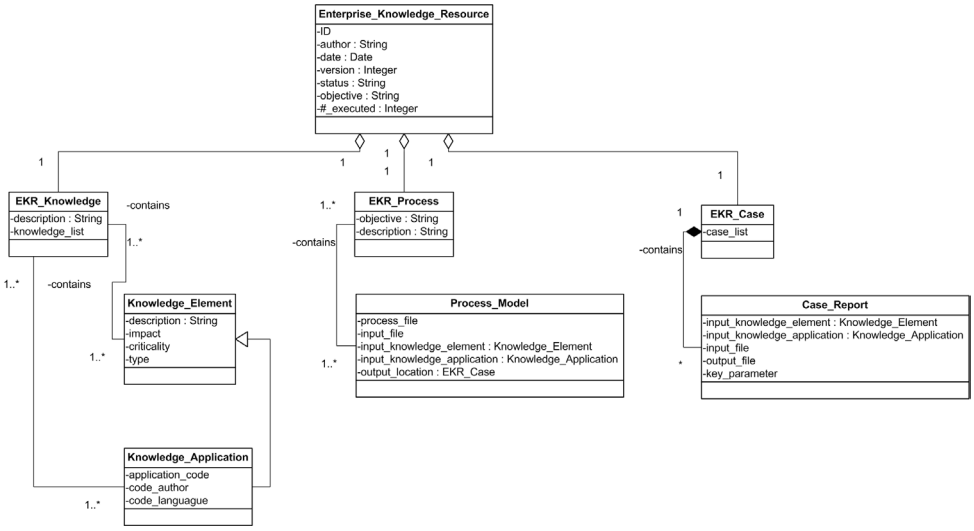

Figure 3.5: UML class diagram of Enterprise Knowledge Resource (Bermell-Garcia *et al.*, 2012)

In research performed jointly by Bermell-Garcia and the dissertation author, the concept of an Enterprise Knowledge Resource has been further researched (Bermell-Garcia *et al.*, 2012; Verhagen *et al.*, 2012). The concept has been reformulated and now expresses a task-oriented container for *knowledge, process*

and *cases*. A Unified Modelling Language (UML) class diagram of an EKR as used in Bermell-Garcia *et al.* (2012) is given in Figure 3.5.

The most important aspects of Figure 3.5 are the EKR class itself and the knowledge, process and case classes. The EKR container class represents the task that needs to be fulfilled, with the attendant attributes such as the task objective. As such, it resembles the task knowledge category of the CommonKADS methodology. Tasks / functions can be identified using functional decomposition (e.g. with IDEF0 or using systems engineering techniques such as functional flow diagrams or functional breakdown structures). The EKR container class incorporates the **EKR_Knowledge**, **EKR_Process** and **EKR_Case** classes.

The **EKR_Knowledge** class of an EKR contains the knowledge elements that are used to perform the specific task associated with an EKR; this can pertain to full KBE applications or to modular knowledge elements that can be combined to fulfil a task. The role of the **EKR_Knowledge** class is therefore to represent and deliver the knowledge that is subsequentially called via the **EKR_Process** class to execute a design or analysis task. In this, the **EKR_Knowledge** class is quite similar to the domain knowledge category of CommonKADS (Schreiber *et al.*, 1999), which represents the main static information and knowledge objects in an application domain.

The **EKR_Process** class contains the workflow of the specific task for which the EKR is set up; it is composed of process elements. In this, it bears some similarity to the task method and inference knowledge categories of the CommonKADS methodology, which are used to control and carry out a sequenced reasoning process on the domain knowledge elements.

The **EKR_Case** class is used to gather and present the design or analysis results that flow out of the use of an EKR (i.e. the execution of a task using a combination of process and knowledge elements). As such, it embodies the generative capability of knowledge.

In summary, the various models presented in this section can be used to fulfil the KLC ontology requirement of functionality. In particular, the EKR approach is suitable for this as it enables the representation of a function and its attributes together with its constituent elements (inputs, activities, outputs).

3.2.2.3 Knowledge Lifecycle Model

The final building block for the KLC ontology is the Knowledge Lifecycle Model as introduced in Section 3.1.

Its most important contributions to the KLC ontology are the concepts of knowledge state and knowledge action. Both have the potential to be incorporated into the lifecycle model as class properties. When included and managed – either by users or by applications – the behaviour of knowledge can be quantified directly from the knowledge-based applications.

3.2.2.4 Summary of KLC ontology building blocks

To recap, the following elements are considered as building blocks for the Knowledge Life Cycle (KLC) ontology:

- Lifecycle elements:
  - PROMISE Semantic Object Model
  - PDW Core Ontology
  - PPR paradigm
- Functional elements:
  - Functional decomposition (IDEF0/CommonKADS)
  - Enterprise Knowledge Resource concept
- Knowledge life cycle model

In Table 3.6, the previously introduced requirements regarding the KLC ontology are related to the building blocks. Which potential contributions from which building blocks address specific KLC ontology requirements?

Table 3.6: KLC ontology requirements in relation with building blocks

KLC ontology – requirements	Building block potential contributions
1. Traceability **i. Visibility:** experts / end users should have the possibility to see the knowledge that feeds into KBS. **ii. Accessibility:** experts / end users should be able to access and work with knowledge. **iii. Retrievability:** experts / end users should be able to effectively search for and find relevant knowledge.	• **PROMISE SOM:** as a semantic object model, the SOM is very adept at facilitating retrievability. Furthermore, the SOM incorporates several classes and properties that aid visibility of knowledge, though its not clear if other classes (e.g. **Resource** and its subclasses) facilitate accessibility of knowledge. • **PDW Core Ontology:** the Core Ontology facilitates traceability of knowledge through its **Design_Traces**, **Description_Object** and **Storage_Object** classes. The latter two are particularly suitable for accessibility of knowledge. The Core Ontology seems less strong in semantic terms, with corresponding doubts regarding the retrievability of knowledge. • **PPR paradigm:** through annotation with **Product**, **Process** and **Resource** classes, concepts can

	theoretically be retrieved and accessed effectively. • **Enterprise Knowledge Resource:** through the **EKR_Knowledge** class of an EKR, knowledge is easy to inspect, use and maintain. An EKR should be annotated to enable retrievability in an enterprise knowledge system. • **IDEF0:** the input and output elements in IDEF0 as well as the resulting hierarchical models of functionality enable traceability, though operationalization of these constructs must be performed. • **Knowledge Lifecycle Model:** the concept of knowledge states can be used to characterize the maturity of knowledge. Furthermore, when earlier versions of knowledge elements are archived, users can inspect the changes in knowledge and learn from the progression.
2. **Functionality & Separation** of tasks and domain knowledge	• **PROMISE SOM:** the SOM includes separate classes that can be used to express tasks (**Activity**) and knowledge (**Resource**, **Physical_Product**). • **PDW Core Ontology:** similar to the SOM, the Core Ontology includes separate classes that can be used to express tasks (**Process_Object** and its subclasses) and knowledge (**Description_Object** and its subclasses). • **IDEF0:** IDEF0 models can be used to represent and decompose functionality, but it is not clear how tasks and domain knowledge are separated. • **CommonKADS:** CommonKADS calls for separation of task, inference and domain knowledge when

	composing a knowledge model. In particular, CommonKADS offers templates for task methods and inference knowledge that improve task knowledge re-use. • **Enterprise Knowledge Resource:** the EKR concept supports the explicit separation of tasks and domain knowledge through its concepts of the EKR class itself, **EKR_Knowledge** and **EKR_Process**.
3. Knowledge management **i.** Across **domains:** users and tools from different domains should work on the basis of consistent data, information and knowledge. **ii.** Across **users / actors** from different perspectives: the KLC ontology should accommodate the perspectives from different users (e.g. production engineer and machine operator in the manufacturing domain), for the different lifecycle stages.	• **PROMISE SOM:** the SOM supports all lifecycle domains and users through its classes and associated properties, in particular the **Resource**, **Life_Cycle_Phase** and **Event** classes. • **PDW Core Ontology:** similar to the PROMISE SOM, the Core Ontology contains classes that support multiple perspectives on knowledge, e.g. the **User** class. • **PPR paradigm:** the **Product**, **Process** and **Resource** classes are useful generic classes that can be used across domains and across users, as shown in Dassault Systemes' commercial systems. However, these classes require more detailed representation to facilitate actual use in a semantic metamodel. • **Enterprise Knowledge Resource:** the EKR concept does not directly support different perspectives on knowledge. This may be achieved indirectly through semantic annotation.

Similar to the research approach adopted in the iPROD project (iProd, 2013), the building blocks have the potential to address the fundamental requirements of having a

a. Capability to represent a usable function (*usability*)

b. Capability to semantically annotate a function in order to be able to find, access, inspect and maintain this function (*usability & maintainability*)

Not all building blocks are fully usable for the development and implementation of the KLC ontology. Though all are considered, the most influential building blocks are the Enterprise Knowledge Resource concept and the PPR paradigm. This will be shown in the following Section.

3.2.3 Research Contribution 2: Knowledge Life Cycle Ontology

The purpose of this Section is to combine the previously introduced building blocks to develop the Knowledge Life Cycle ontology. This Section follows the ontology development methodology steps as identified by Pinto and Martins (2004): specification of the purpose and scope of the ontology in Section 3.2.3.1, followed by conceptualization of the concepts and relationships in Section 3.2.3.2. These concepts and relationships are formalized and implemented into UML in Section 3.2.3.3.

3.2.3.1 Specification: Purpose and Scope of the KLC Ontology

The purpose of the KLC ontology is to enable consistent development of knowledge-based applications that can cope with knowledge change through product life. To achieve this, the model should support the traceability of knowledge through life, such that knowledge and knowledge-based applications can be formalized, used, maintained, reused and retired through life. The model must accommodate users with different perspectives while remaining consistent. Furthermore, it must separate task representation from domain knowledge.

The ontology will be developed to a semi-formal stage: it will be expressed in an artificial formally defined language, but does not incorporate formal semantics, development of predicates, theorems and proofs of properties such as soundness and completeness (Uschold, 1996). Developing the KLC ontology at a semi-formal level is sufficient for the purposes of this dissertation: the case studies that validate the KLC ontology are developed to a proof-of-concept stage and do not require the functionality of a fully formal ontology (see also Section 3.5).

The language adopted for expressing the KLC ontology is the Unified Modeling Language (UML). UML is preferred over OWL and other ontology languages. This choice is backed up by a number of arguments. First, in contrast to most ontology languages, UML modeling supports graphical notation, making it easier to develop, interpret and implement. Furthermore, UML correlates with the desired level of formality of the KLC ontology – the formality and functionality of ontology languages (e.g. predicates and automated reasoning) are not necessary for a semi-formal ontology. Finally, UML is widely adopted and used in

both academia and industry; interpretation and dissemination of the KLC ontology is best supported by UML.

The scope of the KLC ontology is a generic product lifecycle. For practical purposes, its use throughout the remainder of this dissertation will be confined to the aircraft engineering domain, as discussed in Section 1.2.1.1.

3.2.3.2 Conceptualization: Definition of High-Level Concepts and Relationships

To conceptualize the KLC ontology, high-level concepts and relationships from the building blocks have been combined into a single conceptual model. A high-level overview of this model is shown in Figure 3.6. This figure shows that there are two central perspectives in the model. The first is the Product - Process - Resource (PPR) paradigm, as introduced before. The second perspective sees the Enterprise Knowledge Resource as a central concept. Both perspectives are described in more detail below. As an additional note, the EKR itself and the knowledge elements will have the knowledge states and actions from the Knowledge Lifecycle Model as attributes. This enables the quantification of knowledge change over its life.

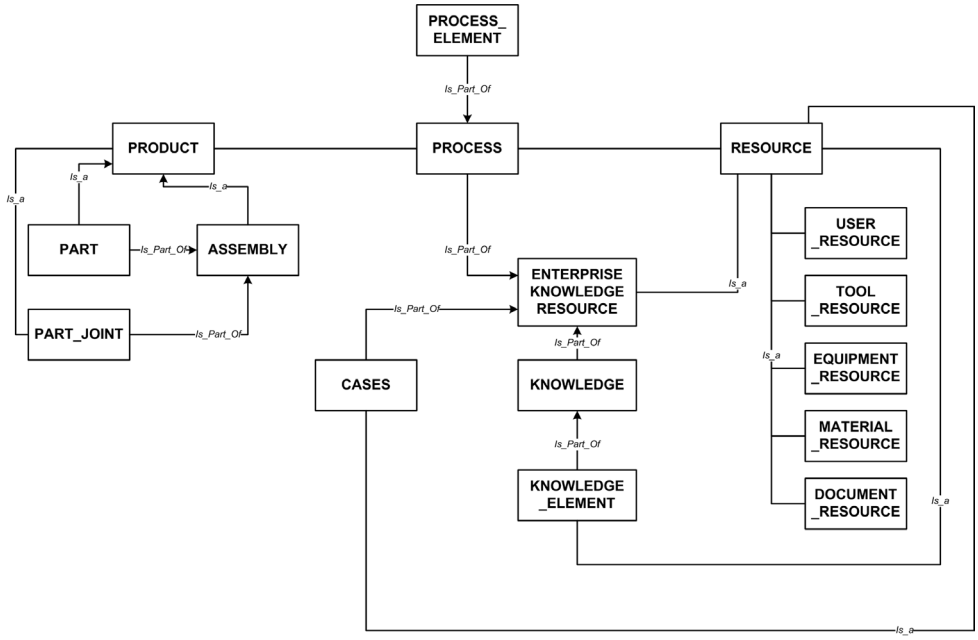

Figure 3.6: High-level concepts and relationships of the KLC ontology

The generic role of an EKR is to provide the capability for a usable function: the ability to (automatically) execute a task, to enable the inspection, review and (possible) revision of the associated knowledge and process elements, and to

enable users to inspect and learn from the outputs of previous runs of the EKR through the collected case reports. The KLC ontology has a similar representation of the EKR container class as introduced in (Bermell-Garcia *et al.*, 2012). However, several changes have been made relative to that incarnation of the EKR concept:

- **Knowledge:** Knowledge contains knowledge elements, which can be seen as a type of Resource. A set of knowledge elements can together constitute the required knowledge for an EKR (task). This relates to explicit, codified knowledge which can include both informal and formal representations of concepts, relationships, assumptions, rules, constraints, rich text descriptions, illustrations, drawings, etc. These elements contain a capability for effective action (and are therefore compliant with the definition of knowledge as adopted in this research), but to harness this capability for action, it is necessary to have a provision in place to further organize and actively use the knowledge elements. This provision is provided through the Process concept.
- **Process:** The Process concept consists of process elements. In a similar manner to the knowledge concept, a set of process elements (e.g. activities) can be combined to form the process that is required for an EKR. In other words, the Process concept contains the workflow of the specific task for which the EKR is set up.. In the EKR concept, the Process class can contain a sequence of process elements (activities), such as importing an input file, calling a knowledge element or a self-contained knowledge application, executing the knowledge application, collecting and passing on the outputs, calling another knowledge application, executing this knowledge application, and so on. The workflow can be automated, though this of course depends on the implementation. The role of the Process concepts is to represent and enable an “end-to-end” engineering task, preferably in an automated fashion (i.e. without the requirement of user intervention). In combination with knowledge elements, the process elements can be used to create a fully automated, white-box design or analysis process. Users can inspect the steps in the design or analysis process, and can see the associated knowledge through the related knowledge element(s).
- **Case:** The Case concept is used to gather and present the design or analysis results that flow out of the use of an EKR (i.e. the execution of a task using a combination of process and knowledge elements). As such, it embodies the generative capability of knowledge. Every time a specific EKR is executed, a case report can be generated. This report includes an overview of the inputs that were used for the analysis, such as CAD drawings or Excel files. Furthermore, the knowledge elements that were

used when running an EKR instance can be listed. Finally, the output files generated by the analysis (e.g. stress distribution graphs or cost estimations) can be listed. All of these elements are directly accessible through the case report; for instance, a user can directly go to and inspect the knowledge that was used for a specific analysis run. When an EKR is run multiple times, for instance with multiple sets of different inputs, the results are gathered in a set of case reports. This enables the subsequent inspection of analysis results, but also opens up the opportunity to further analyze the results themselves. Case metadata can also be automatically assigned, which enables consistent categorization and easier search and retrieval of historical analysis results.

To support the use of EKRs in practice, the aforementioned PPR paradigm is the guiding principle in the development of a semantic metamodel that can be used to annotate EKRs for systematic storing and indexing into a digital enterprise repository. This facilitates the reuse, sharing and maintenance of EKRs, as well as the knowledge upon which EKRs operate. The PPR paradigm is met by including Process, Resource and Product concepts, as shown in Figure 3.6. The Resource concept encapsulates a relatively large number of subconcepts, for instance User and Tool. Finally, the Product concept can be used to represent individual parts and, when joined, assemblies. This is of importance to represent the Product Breakdown Structures often used in aircraft engineering.

With respect to the relationships between the various concepts, Figure 3.6 shows only the most important relations between high-level concepts, such that the figure retains some clarity. At lower levels, concepts may also share relations, though most are not depicted here (e.g. Knowledge_Element will be related to a User_Resource and Document_Resource).

3.2.3.3 Formalization and Implementation: Transformation of Conceptual Model into a Semi-Formal UML Ontology

The next step towards a Knowledge Life Cycle ontology is the transformation of the conceptual model into a semi-formal ontology. As mentioned, the chosen language for expressing the semi-formal model is UML. The following activities have been performed:

- Transformation of concepts into classes with their most important attributes
- Formalization of class relationships, including aspects such as multiplicity and type
- Implementation of classes and relationships in UML

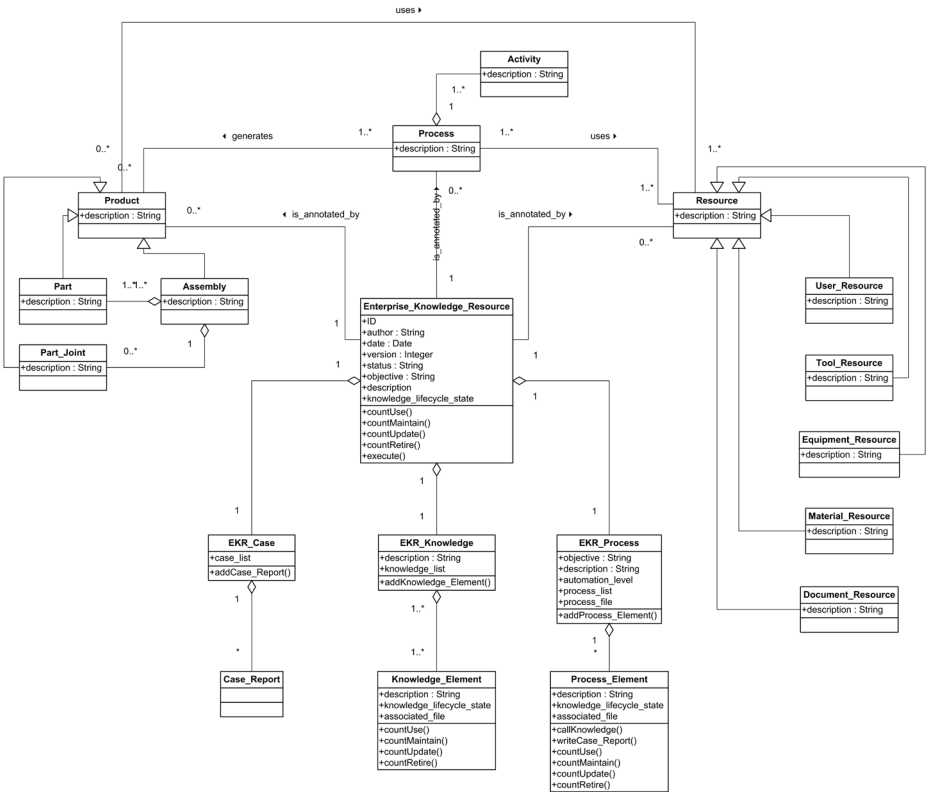

Figure 3.7: UML class diagram of KLC ontology

The resulting UML model of the KLC ontology is presented in Figure 3.7. The following main classes and attributes are further explained:

- **Enterprise_Knowledge_Resource:** this class is the central way of representing functionality into the KLC ontology. It contains the **EKR_Knowledge**, **EKR_Process** and **EKR_Case** classes. Its attributes include basic metadata (e.g. author, version) and a knowledge_lifecycle_state attribute. This attribute can be used to describe the instantaneous state of an EKR. For instance, if a user makes a change to the semantic annotation of an EKR, the state jumps to 'Maintain'. When a user has finished making changes (possibly subject to a validation process), the state changes back to 'Use'. The possible states are taken from the knowledge lifecycle model and include 'Create' (initate instance), 'Use' (completed/validated instance), 'Maintain' (context of instance is changed), 'Update' (content of instance is changed), 'Retire' (instance is binned or deleted). To quantify knowledge change, operations that count changes in state (i.e. knowledge actions) are added to the EKR

class. A UML class diagram for Enterprise Knowledge Resource as implemented within the KLC ontology is given in Figure 3.8.

- **EKR_Knowledge**: the **EKR_Knowledge** class contributes to the EKR class. Its main attribute is knowledge_list, which can serve to collect knowledge elements that are used for a particular EKR. The **EKR_Knowledge** class has one main subclass: **Knowledge_Element**. Instances of **Knowledge_Element** can be combined with **Process_Element** instances to configure a knowledge-based application. Like the EKR class, the **Knowledge_Element** class has an attribute to keep track of the knowledge lifecycle state as well as operations that can count changes in state (i.e. knowledge actions). The Knowledge class (and its subclass) can be annotated by **Product**, **Process** and **Resource**, and their respective subclasses.
- **EKR_Process:** the **EKR_Process** class contributes to the EKR class. It has a number of attributes, for instance for objective and description. Notably, the level of process automation can be expressed using the relevant attribute. **EKR_Process** has one main subclass: **Process_Element**. One or more process elements can be combined to form a process. Like the EKR class, the **Process_Element** class has an attribute to keep track of the knowledge lifecycle state as well as operations that can count changes in state (i.e. knowledge actions). Further operations are added to model the potential use of knowledge elements in a process (callKnowledge()) and to write the results of process execution into a case report (writeCase_Report()). In contrast to Figure 3.6, the process representation of the EKR (**EKR_Process)** has been separated relative to the generic **Process** class (see below).
- **EKR_Case:** the **EKR_Case** class contains the case history of EKR use. Every time an EKR is run, an instance of the subclass **Case_Report** is generated and populated within the **EKR_Case** class. Instances of **Case_Report** contain an overview of the knowledge and process elements that were used in that particular EKR run. This enables backward traceability of the results of an EKR run.
- **Product:** the **Product** class can be used to generate product-oriented views and can be used to annotate EKRs and its subclasses. An association has been made between the **Product** and **Enterprise_Knowledge_Resource** classes; similar associations exist between **Product** and the EKR subclasses, but these are not shown. To annotate effectively, the **Product** class can be extended into domain-specific class hierarchies. This will be done in Chapters 4-6, where relevant Product Breakdown Structures (PBS) are developed for the design, manufacturing and maintenance domains.

- **Process:** The **Process** class is part of the PPR semantic metamodel that can be used to annotate EKRs. An association has consequently been made between the **Process** and **Enterprise_Knowledge_Resource** classes. To achieve annotation, **Process** can be extended into a domain-specific class hierarchy – this will be shown in Chapters 4-6.
- **Resource:** the **Resource** class can be seen as the superclass of the **Enterprise_Knowledge_Resource** class, though this relation is not given in Figure 3.7. Besides this, the class contains users, tools, equipment, materials and documents as subclasses. It can be extended into domain-specific class hierarchies for annotation of EKRs. As before, this will be shown in Chapters 4-6.

The main concept relationships can be seen in Figure 3.7; for completeness' sake they are also given in Table 3.7.

Table 3.7: Relationships between main concepts of KLC ontology

Class 1	Class 2	Relation (name)	Relation (type)
Process	**Product**	*generates*	Association
Process	**Resource**	*uses*	Association
Product	**Resource**	*uses*	Association
Enterprise_Knowledge_Resource	**Product**	*is_annotated_by*	Association
Enterprise_Knowledge_Resource	**Process**	*is_annotated_by*	Association
Enterprise_Knowledge_Resource	**Resource**	*is_annotated_by*	Association
Assembly	**Product**	*is-a*	Generalization
Part	**Product**	*is-a*	Generalization
Part_Joint	**Product**	*is-a*	Generalization
Assembly	**Part**	*contains*	Aggregation
Assembly	**Part_Joint**	*contains*	Aggregation
User_Resource	**Resource**	*is-a*	Generalization
Tool_Resource	**Resource**	*is-a*	Generalization
Equipment_Resource	**Resource**	*is-a*	Generalization
Material_Resource	**Resource**	*is-a*	Generalization
Document_Resource	**Resource**	*is-a*	Generalization
Enterprise_Knowledge_Resource	**EKR_Process**	*contains*	Aggregation
Enterprise_Knowledge_Resource	**EKR_Knowledge**	*contains*	Aggregation
Enterprise_Knowledge_Resource	**EKR_Case**	*contains*	Aggregation
EKR_Knowledge	**Knowledge_Element**	*contains*	Aggregation
EKR_Process	**Process_Element**	*Contains*	Aggregation
EKR_Case	**Case_Report**	*contains*	Aggregation

The Enterprise Knowledge Resource concept remains the central part of the KLC ontology. It is represented separately in Figure 3.8 and contains the **Enterprise_Knowledge_Resource** class and its subclasses.

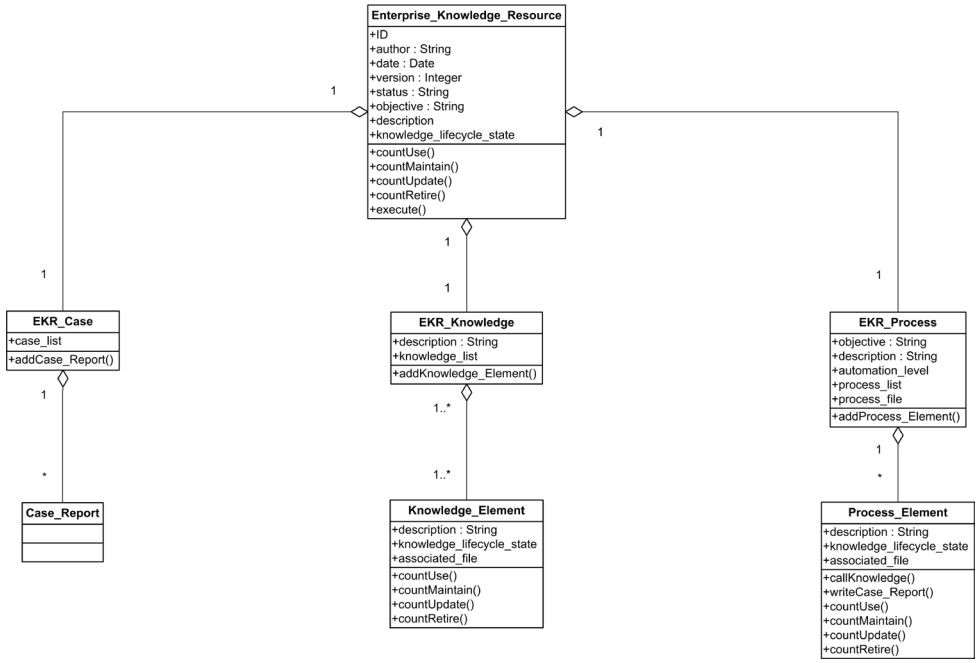

Figure 3.8: UML class diagram of Enterprise Knowledge Resource as implemented in the KLC ontology

The EKR concept allows for the representation of tasks / functions and the inputs (knowledge elements), transformations (process elements) and outputs (case reports) that are associated with a task. Through association with the PPR classes of the KLC ontology, EKRs can be semantically annotated to allow traceability (including visibility, accessability, retrievability) and knowledge management. Practical examples of this are given in Chapters 4-6.

3.2.4 Concluding Remarks

The ontology requirements as introduced in Section 3.2.1 have been met by development of the KLC ontology. This is further explained in Table 3-8, which outlines the requirements and characteristics of the developed ontology.

The KLC ontology has been developed to a semi-formal level and has been implemented in UML. Conversion into a dedicated ontology language (e.g. OWL) is possible – see for instance Verhagen and Curran (2011) – but not necessary for the purposes of this dissertation.

To summarize, the research questions mentioned in the introduction of Section 3.2 have been addressed by introducing the EKR and PPR concepts and relationships and formalizing them into the KLC ontology. Though strongly rooted in earlier research, the ontology is a novel contribution to theory. It can be used

to fulfil the usability and maintainability requirements associated with knowledge change in knowledge-based applications.

Table 3-8: KLC ontology requirements versus functionality

KLC ontology – requirements	KLC ontology - characteristics
1. Traceability **i. Visibility:** experts / end users should have the possibility to see the knowledge that feeds into KBS. **ii. Accessibility:** experts / end users should be able to access and work with knowledge. **iii. Retrievability:** experts / end users should be able to effectively search for and find relevant knowledge.	• **Enterprise Knowledge Resource:** through the "Knowledge" element of an EKR, knowledge is easy to inspect, use and maintain. An EKR can be annotated by **Product**, **Process** and **Resource** (and their subclass hierarchies) to enable retrievability in an enterprise knowledge system. • **Case:** the **Case** class offers the possibility to inspect historical results of running an EKR, including the knowledge elements and process models that were used in a particular EKR execution run. • **Knowledge Lifecycle Model attributes:** the concept of knowledge states can be used to characterize the maturity of knowledge. Furthermore, when earlier versions of knowledge elements are archived, users can inspect the changes in knowledge and learn from the progression.
2. Functionality & Separation of tasks and domain knowledge	• **Enterprise Knowledge Resource:** through the **Process** and **Knowledge** classes, tasks and domain knowledge can be separated, while remaining part of a functional whole. o **Process:** the **Process** class includes subclasses and attributes to model tasks. To achieve tasks, knowledge can be called from the **Knowledge** class. o **Knowledge:** the **Knowledge** class contains individual knowledge elements and knowledge applications that

	can be called via the **Process** class to perform a task.
3. Knowledge management **i.** Across **domains:** users and tools from different domains should work on the basis of consistent data, information and knowledge. **ii.** Across **users / actors** from different perspectives: the KLC ontology should accommodate the perspectives from different users (e.g. production engineer and machine operator in the manufacturing domain), for the different lifecycle stages.	• **Enterprise Knowledge Resource:** through annotation with **Product**, **Process**, **Resource** and its subclasses, an EKR can be accessible across domains and across users. When expressed in the OWL-DL format, a common vocabulary – including equivalent terms – can be declared to accommodate different domain and user perspectives. • **Resource_User:** this class can be used specifically to represent the various users along the product lifecycle. • **Knowledge Lifecycle Model attributes:** the knowledge state and action attributes associated with **Enterprise_Knowledge_Resource**, **Knowledge** and **Knowledge_Element** can be used to express the status and maturity of knowledge, making it possible to work on the basis of consistent knowledge.

3.3 The KNOMAD Methodology for Supporting KBS Development incorporating Knowledge Change

The preceding Sections have discussed the models necessary for characterising, measuring and facilitating knowledge change. However, these models must be accompanied by a 'how-to': how can the models be applied for the development of knowledge-based applications? This is reflected in the third research objective expressed in Section 1.2: methodology development. The following research questions are associated with this objective:

- How can knowledge change be accommodated during knowledge-based application development?
    - Which steps are required?

To answer these research questions, existing methodologies will be reviewed for their strengths and weaknesses. The identified shortcomings are the basis for proposing a methodology for the development of knowledge-based application that offer better usability and maintainability through incorporation of knowledge change.

3.3.1 State of the Art in Methodologies for KBS development

A number of methodologies have been developed over the years to guide the development of knowledge-based systems. Here, two prominent methodologies are discussed: CommonKADS (Common Knowledge Acquisition and Documentation Structuring) and MOKA (Methodology and software tools Oriented to Knowledge-based engineering Applications).

CommonKADS focuses on the development of Knowledge-based Systems (KBS). It consists of steps, guidelines, models and templates. The main elements of CommonKADS are given in Figure 3.9. Three layers can be distinguished. The top layer incorporates the organisation, task and agent models and is preparatory in nature. In this layer, opportunities for KBS development are identified, a KBS project is scoped and initial analysis is performed to map the available knowledge and knowledge users. Potential solution directions are distinguished. The middle layer uses the inputs from the top layers models and takes the project further by detailed analysis of knowledge, resulting in the knowledge model. This model encompasses task, inference and domain knowledge and preferably uses knowledge and task templates to promote re-use. The communication model complements the knowledge model by considering the use of knowledge and information within the organisation and between users. Finally, the bottom layer comprises the design model: the KBS is designed and implemented. The focus is

on the technological aspects of KBS development. Schreiber *et al.* (1999) further detail the CommonKADS layers and associated models.

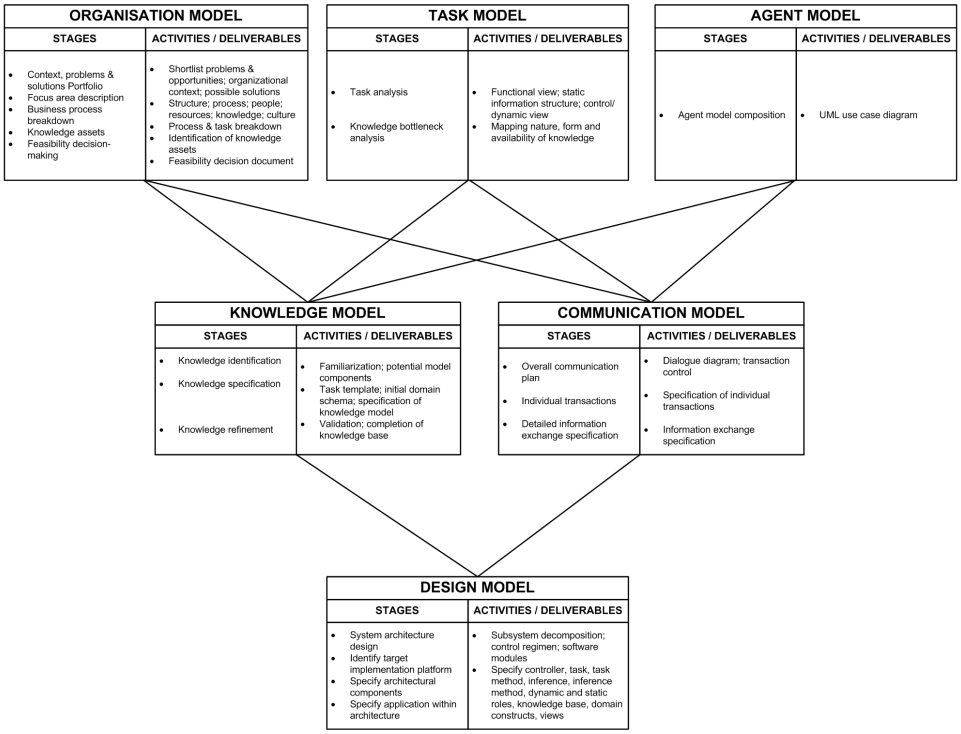

Figure 3.9: CommonKADS methodology overview

From a design engineering perspective, a number of methodologies are available to support the development of KBE applications (Lovett *et al.*, 2000; Stokes, 2001; Curran *et al.*, 2010; Verhagen *et al.*, 2012). By far the most well-known of these is the Methodology and software tools Oriented to Knowledge-based engineering Applications, or MOKA. This methodology, based on eight KBE life-cycle steps, is designed to take a project from inception towards industrialization and actual use (Stokes, 2001). The KBE System Lifecycle is given in Figure 3.10 and is a process view of the lifecycle stages of a KBE system. The first stages (Identify; Justify) are similar to the Organisation model of CommonKADS; the purpose of these stages is to identify, analyse and scope opportunities for KBE development. The next stages are Capture and Formalise; these stages are similar to the Task and Knowledge models of the CommonKADS methodology. Their purpose is to capture and model the knowledge and activities that are associated with the KBE project. The remaining stages are Package, Distribute, Introduce and Use: these are similar to the Design Model of CommonKADS, but with added emphasis on the actual use of the KBE system. Another difference with

CommonKADS is the explicit consideration of KBE system maintenance as an alternative to a new system after the Justify step, and the acknowledgement through a feedback loop that the Use stage may trigger a cycle of maintenance. However, though maintenance is explicitly identified, MOKA does not offer guidelines for maintenance of knowledge-based engineering systems other than to repeat the whole KBE System Lifecycle process when changes in knowledge occur. MOKA does not incorporate a method to characterise the quantity and frequency of knowledge change. Also, for which types and at what quantity of change is any threshold to trigger maintenance to be fired? Furthermore, it is not clear how existing models built in the Capture and Formalise stages are to be adapted given new or changed knowledge. Moreover, it is not clear how these changes are to be propagated into the packaged KBE system while maintaining knowledge base consistency and reliability.

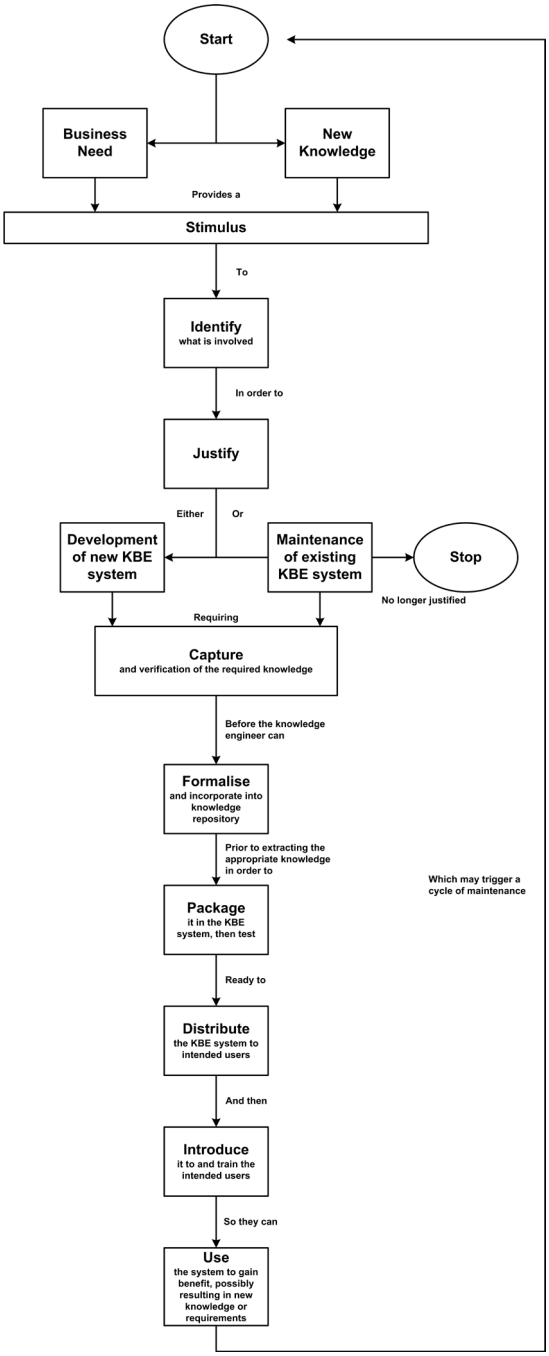

Figure 3.10: KBE System Lifecycle (adapted from Oldham *et al.* (1998))

The Capture and Formalise stages are the central contribution of the MOKA methodology. The centrepieces of these stages are the Informal and Formal

MOKA models. The informal model consists of so-called ICARE forms, where the acronym stands for Illustrations, Constraints, Activities, Rules and Entities. These forms can be used to decompose and store individual knowledge elements. Subsequently, these elements can be linked to create a structured web of knowledge elements that together make up a representation of the problem domain to which users from multiple viewpoints can relate. When the problem knowledge has been converted into a structured representation, the next step is to formalize this knowledge in order to represent knowledge in a form that is acceptable to knowledge and software engineers and suitable for subsequent development of a KBE application. The formal model uses MML (Moka Modelling Language, an adaptation of Unified Modelling Language (UML)) to classify and structure the ICARE informal model elements, which are translated into formal Product and Process models. The main elements of the MOKA methodology are illustrated in Figure 3.11: the KBE system lifecycle, the Informal model (as illustrated by an ICARE form) and the Formal model (as represented by an MML structure). A more in-depth discussion of MOKA can be found in Stokes (2001).

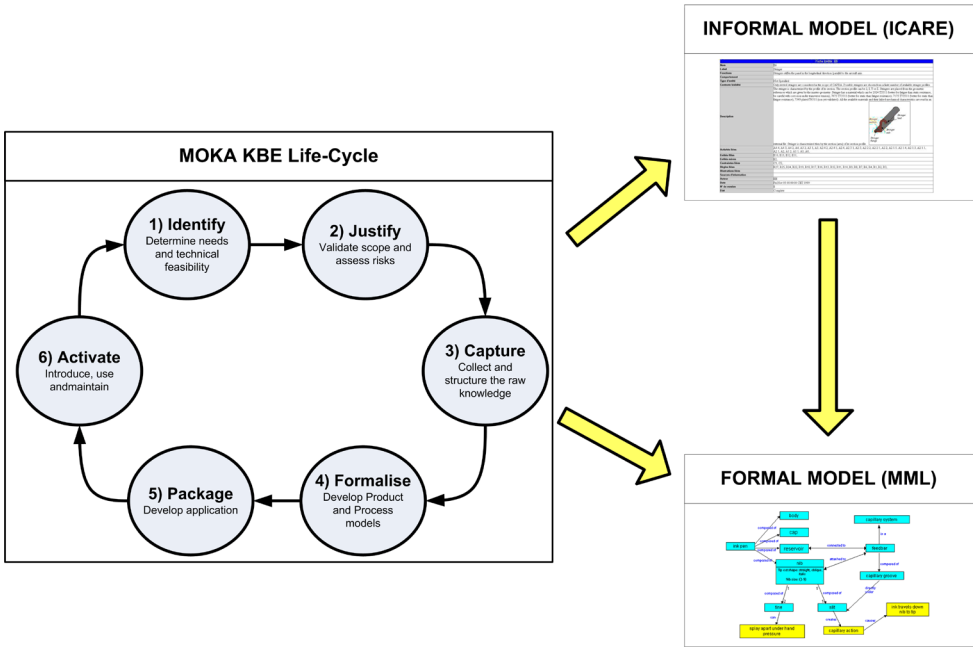

Figure 3.11: MOKA methodology elements

3.3.2 Shortcomings of Existing Methodologies and Associated Research Requirements

Both CommonKADS and MOKA acknowledge the change of knowledge over time and the associated need for application maintenance to ensure usability. Schreiber *et al.* (1999) mention that "the basic idea underlying the CommonKADS model suite is that it provides a correct and full view of the status of application development…Because knowledge is not static but changes over time, the process is best seen as continuous development. Maintenance of the knowledge model is thus not essentially different from its development. The main difference is that … it is often done by other people… If the knowledge model of an application is good and the domain is stable, one can expect the majority of maintenance to be concerned with activity 'complete the knowledge bases'. Typically, sets of rule instances will need to be updated, because knowledge tends to evolve over time" (page 184). Similarly, the MOKA handbook mentions that "many companies that use KBE are faced with a dilemma. Existing applications need to be updated or improved. … Generally, it is recommended that you should introduce any changes in the form of new or modified ICARE forms and then adjust the MOKA Formal Models before changing the KBE application files" (page 263). Note that the change or 'evolution' of knowledge is stated as a given. A major and essential drawback of these methodologies is that they *do not investigate the nature and consequences of knowledge change in any depth*. Instead of enabling the development of change-compliant solutions, both methodologies advise to go through all steps of the methodology again – a potentially costly exercise without many guidelines regarding how to deal with and/or change the existing application. Through their steps and models, both major methodologies are not able to directly cope with knowledge change in knowledge-based applications. When this situation occurs, rework is necessary.

The MOKA and CommonKADS methodologies can further be compared to methodology requirements such as flexibility, scalability, extendibility (Colledani *et al.*, 2008), completeness and applicability. As a result, other shortcomings of these methodologies can be identified. CommonKADS is fit for complex cases with requirements on formal specification and re-use. Fairly prescriptive and detailed guidelines and templates are available to support knowledge engineers, which improves completeness and replicability, but decreases flexibility. Moreover, the CommonKADS methodology as described by Schreiber *et al.* (1999) lacks specificity for the engineering domain: there are very few knowledge and task templates for engineering tasks.

The main focus of MOKA lies with the 'Capture' and 'Formalize' steps of the KBE life-cycle. Curran *et al.* (2010) have identified that MOKA has the following drawbacks as a result: a focus on knowledge engineering support rather than end

user support, and a lack of provisions for knowledge transparency and accessibility.

Another major shortcoming of existing methodologies is a lack of methodology adherence (Curran *et al.*, 2010; Verhagen *et al.*, 2012). In particular, a review of the KBE domain (Verhagen *et al.*, 2012) has shown that 81% of the sample papers did not adhere to or even mention any specific methodology when developing a KBE application. Methodology adherence cannot be forced, but existing methodologies have seemingly failed to make a large impact.

A final shortcoming is the lack of quantitative evaluation of knowledge-based application development. The MOKA and CommonKADS methodologies do recommend considerable attention for the identification, evaluation and justification of a business case for the development of a knowledge-based system. This is an activity carried out *before* the development of KBS. Both methodologies do not follow this up with quantitative evaluation of the KBS development *after* completion of projects. Such (quantitative) evaluation is necessary to draw lessons from KBS projects.

Given these shortcomings, a research challenge can be distilled: develop an improved methodology for the development of knowledge-based applications which must cope with changing knowledge. Such a methodology would have to meet the following requirements:

- Be able to facilitate knowledge change.
- Include an approach for knowledge capture, formalization, *use and maintenance*.
- Be sufficiently simple and straightforward to 'invite' use.
- Include steps / guidelines for the assessment of KBS performance.

3.3.3 Research Contribution 3: KNOMAD Methodology

To meet the improved methodology research challenge, the KNOMAD methodology Curran *et al.* (2010) is introduced. KNOMAD consists of the following main steps: **(K)nowledge Capture & Identification of Knowledge Change, (N)ormalisation, (O)rganisation, (M)odelling & Implementation, (A)nalysis** and **(D)elivery**.

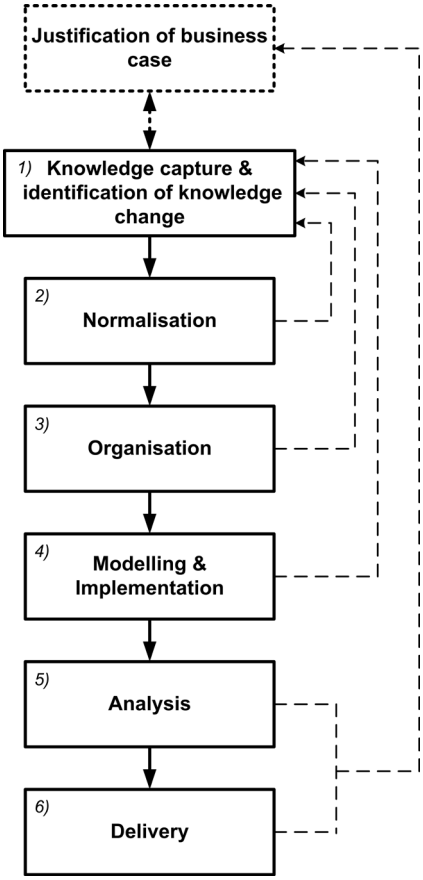

Figure 3.12: KNOMAD methodology overview

The main steps are given in Figure 3.12. As can be observed, the first activity before initiating the main steps is to justify the business case. This is similar to the Identify & Justify steps of MOKA, which have the objective to identify, analyse and scope opportunities for KBE development. In the justification of the business case, it can be judged whether the process / task is suitable for a knowledge-based approach; as mentioned before, Van der Velden *et al.* (2012) provide a first attempt in this direction by considering process complexity. The scope, objectives and context of the project can be established. Furthermore, business metrics (e.g. ROI) can be identified and analysed to judge the business case (Van Dijk *et al.*, 2012). Some first forays into knowledge capture may be necessary to back up the justification effort, as indicated by the dotted line in Figure 3.12.

When a development effort is warranted, the main steps of the KNOMAD methodology can be applied. They are explained in the following sections.

3.3.3.1 Knowledge Capture & Identification of Knowledge Change

This first step comprises two substeps: 1) capturing 'raw' knowledge involved in the project, and 2) characterising and identifying any knowledge change.

Knowledge acquisition techniques can be applied to capture the knowledge. A wide range of knowledge acquisition techniques is available; interviews, process mapping, laddering, state diagram mapping, concept mapping, observation and commentating are some examples (Schreiber *et al.*, 1999; Stokes, 2001; Milton, 2007). The captured knowledge has to be documented to support its use in the following steps.

Following capture of the knowledge, the Knowledge Lifecycle Model can be applied to characterise and quantify knowledge change. The knowledge actions can be operationalised and measured for the specific project under consideration. Based upon the observed behaviour of the knowledge, an assessment of stability over time can be made, which informs a choice for a specific development method. If knowledge is found to be changing, the development process can be based on the further steps of the KNOMAD methodology and use the KLC ontology to enable an ontology-based approach towards knowledge lifecycle management.

3.3.3.2 Normalisation

In the Normalisation phase, the raw knowledge obtained in the knowledge capture phase is subjected to quality control and normalization. The objective of the normalization phase is to achieve a sufficiently high quality level of knowledge content to enable the seamless use of knowledge in subsequent organization, modelling and analysis. To achieve this, two distinct activities are performed. First, the knowledge is checked against applicable quality criteria. The following general quality criteria are recognized:

- **Traceability:** knowledge must be traceable to its source and must be traceable over various iterations. The knowledge state and any performed knowledge actions on the knowledge element must be recorded.
- **Ownership:** a knowledge element has to be tied to an owner. This owner takes responsibility for the accuracy and reliability of the knowledge. This is a prerequisite for efficient knowledge management, especially when tacit knowledge is involved. It can have significant benefits in later stages (organization, modelling, analysis) if necessary knowledge needs further explanation, as the owner can easily be contacted.
- **Accuracy:** is the captured knowledge accurate enough for subsequent use in modeling and analysis? The modelling and analysis of parameters puts requirements on accuracy, as significant variation or uncertainty in design parameters may prevent model and analysis resolution.

- **Reliability:** how reliable is the captured knowledge? Particularly during early stage design, parameters have a tendency of being 'guesstimates', or they are founded on unchecked assumptions. If knowledge is highly dynamic, this should be taken into account during subsequent development steps. In fact, the speed of change may be so high that the knowledge in question is deemed too immature for further modelling and analysis.

The second activity is normalization according to *standards*. The range of available standards to normalize against is considerable; the applicable standards for normalization depend on the context. Standards that must be set and/or adhered to are for instance programming language, units and regulations.

3.3.3.3 Organization

The organization of knowledge is an essential step towards knowledge utilization in knowledge-based applications. Its purpose is to provide a knowledge structure to organize the captured knowledge. Furthermore, it can serve as a semantic backbone for the knowledge-based application. To achieve this, it is necessary to construct a domain-specific set of concepts and relationships: a domain ontology. The ontology forms the semantic knowledge structure that can be used to annotate the knowledge-based application and/or enterprise knowledge resources (see also step 4: Modelling). The domain ontology can be linked with the KLC ontology by extending the **Product**, **Process** and **Resource** classes into domain-specific class hierarchies that include all relevant domain concepts. Practical examples of this are given in Chapters 4-6. Furthermore, the domain ontology is a way to represent multiple viewpoints of domain experts on the same domain. It can be used to facilitate end user understanding of the knowledge-based application, by offering 'entry points' into the application through familiar concepts rather than high-level abstractions such as **Process** and **Resource**. Additional domain understanding gained during this organization step can be fed back into the knowledge identification step.

3.3.3.4 Modelling & Implementation

The next step in the revised KNOMAD methodology is the modelling of knowledge-intensive processes. To achieve this, the CommonKADS Knowledge and Design models or the MOKA informal and formal models can be used. However, to facilitate knowledge change in knowledge-based applications, it is advised to apply the KLC ontology and in particular the Enterprise Knowledge Resource concept to model tasks and associated inputs and outputs in the form of knowledge, process models and cases. The modelled EKR(s) can be annotated using the domain ontology, as mentioned above. Annotation enables stakeholders from various disciplines to (automatically) access and retrieve the necessary knowledge for use and maintenance. Again, practical examples of this are given in

Chapters 4-6. The modelling step can also give rise to revised understanding of knowledge, which may lead back to the knowledge identification step.

After modelling, the next step is to implement the models and develop the knowledge-based application. Rather than reinventing the wheel, adherence to the guidelines for implementation described by the CommonKADS methodology is recommended. Schreiber *et al.* (1999) describe these guidelines in chapter 11: Designing Knowledge Systems (page 271-294). The main steps for implementation are to design a system architecture, identify the target implementation platform, specify architectural components and specify the application within the architecture. It must be noted that the EKR concept should be incorporated in the implementation too, such that structure-preserving design principles are obeyed. This improves system maintainability (Schreiber *et al.*, 1999). The **Knowledge** and **Knowledge_Element** classes of the EKR concept can subsequently be employed to incorporate knowledge change through the use and maintenance of knowledge.

3.3.3.5 Analysis

As part of the implementation step, proof-of-concept versions of Enterprise Knowledge Resources (EKRs) can be developed to test and analyze the functionality of the individual EKRs and the overall system. This analysis can comprise qualitative and quantitative aspects. For instance, requirements compliance would be a qualitative check on the performance of the system. Quantitative analysis can be performed by defining and measuring Key Performance Indicators (KPIs) for the developed system, for instance computing time or the time taken to find and access an EKR.

Furthermore, once an EKR is used in an operational setting, the resulting body of case reports can be subjected to analysis approaches such as Design of Experiments (DoE).

3.3.3.6 Delivery

As a final step, the developed application is delivered to the stakeholders and resource implications are evaluated. A review and acceptance check should be performed in which the developed solution is compared with the requirements to determine the solution validity and suitability.

3.3.4 Concluding Remarks

KNOMAD has been introduced as a methodology for the development of knowledge-based applications that must cope with changing knowledge. The requirements for an improved methodology are briefly revisited. KNOMAD answers to these requirements by including steps and guidelines to capture, formalize, use and maintain knowledge. Furthermore, the critical aspect of knowledge change (and associated maintenance) is accounted for by the characterisation and analysis of knowledge change at the start of the KNOMAD

process. Furthermore, inclusion of the KLC ontology within the modelling step of the KNOMAD methodology and development of a domain ontology in the organization step are specifically advised to ensure usability and maintainability. The domain ontology can be used to semantically annotate the knowledge-based application, so that it can be searched for and retrieved easily. The KLC ontology can be used to model and implement task-specific knowledge, such that this can be maintained when it changes.

The KNOMAD steps have been substantiated to a limited degree in the preceding section. In Chapters 4-6, case studies are presented that use the KNOMAD methodology as a guide for development. These case studies provide additional insight into the application of KNOMAD in multiple engineering disciplines.

3.4 Discussion of Contributions

To summarize, the following three contributions to theory have been developed:

1) **Knowledge Life Cycle model:** a model for the characterisation and quantification of knowledge change has been developed. The model is centred on the concepts of *knowledge state* and *knowledge action*.
2) **KLC ontology:** an ontology has been developed that can be used to model and implement knowledge-based applications, while addressing usability and maintainability requirements. The ontology uses the Enterprise Knowledge Resource concept to represent tasks and associated knowledge, processes and cases. The Product-Process-Resource paradigm is used as the basis for a generic semantic metamodel. This metamodel can serve as a starting point for further development of domain-specific semantic metamodels.
3) **KNOMAD:** a methodology has been proposed to support the development of knowledge-based applications that must cope with knowledge change. The methodology consists of six steps. It advises the use of the Knowledge Lifecycle model and KLC ontology to characterise, model, implement and analyse knowledge-based applications.

3.4.1 Discussion of the Knowledge Lifecycle Model

The Knowledge Lifecycle Model aims to address the following questions within the context of the first research objective – knowledge life cycle modelling:

- Which concepts and relationships are required to characterise the change of explicit knowledge within and throughout the aircraft lifecycle phases?

It uses the knowledge state and action concepts to characterise the change of explicit knowledge. However, further questions remain regarding the change of

knowledge within specific aircraft lifecycle phases and regarding the quantification of knowledge change:

- How does explicit knowledge change within specific phases of the aircraft lifecycle?
- Is change of explicit knowledge quantifiable?

These research questions are still open at this point. Empirical study regarding the characterisation and quantification of knowledge change is not performed in this theory-oriented Chapter. Until the case study results have been presented, the following **assumptions** will apply:

1) Knowledge changes throughout aircraft life.
2) Knowledge changes while in individual aircraft life phases.

With respect to the characterisation of knowledge change, the case studies (Chapters 4-6) will each include a short discussion of knowledge change for the specific case. With respect to the quantification of knowledge change, the maintenance case study (Chapter 6) will include an analysis of Airworthiness Directives to see whether the Knowledge Lifecycle Model's concepts can be applied to meaningfully quantify knowledge change in practice.

3.4.2 Discussion of the Knowledge Lifecycle Ontology

The Knowledge Lifecycle Ontology aims to address the second research objective expressed in Section 1.2 by providing a model-based approach for the support of knowledge change within knowledge-based applications for specific aircraft lifecycle phases. The following research questions are addressed:

- Which concepts support the consistent formalization, use and maintenance of changing knowledge throughout the aircraft lifecycle?
- How can knowledge change be accommodated during knowledge-based application development?
    - Which models are required and how do these models help to accommodate knowledge change?

The answer to these questions is partially contained in Table 3-8, which outlines the requirements and characteristics of the KLC ontology. The main concepts used in the KLC ontology for consistent formalization, use and maintenance of changing knowledge are the Enterprise Knowledge Resource concept and the Product-Process-Resource paradigm. In combination, these two elements provide a means to ensure usability and maintainability of knowledge in knowledge-based systems.

The KLC ontology draws inspiration from many existing models, including the PROMISE Semantic Object Model, the PDW Core Ontology, CommonKADS'

Knowledge Model, Bermell-Garcia's concept of Enterprise Knowledge Resources and the Product-Process-Resource paradigm. The KLC ontology sets itself apart from these models in four major ways:

- The ontology's central concept of EKRs can be used as central element in modelling *and implementation* (in contrast to CommonKADS, where the knowledge and communication model is preparation for the implementation-oriented design model). EKRs are a means to achieve structure-preserving implementation, which is beneficial for maintainability;
- The KLC ontology uses the EKR concept to represent individual tasks; this allows for modular and contained development of knowledge-based systems, but requires attention to functional decomposition and interface management. Knowledge associated with specific tasks can be inspected, used and maintained when necessary;
- EKRs can be consistently annotated using the PPR classes, allowing for improved traceability and accessibility;
- Outputs are systematically stored (using the **EKR_Case** class).

The KLC ontology allows for consistent formalization, use and maintenance of knowledge and knowledge-based applications. To validate this assertion and the KLC ontology itself, the ontology will be applied to the development of three knowledge-based applications. This is described in more detail in Chapters 4-6, where three case studies are developed for the design, manufacturing and maintenance phases of the aircraft lifecycle.

3.4.3 Discussion of the KNOMAD methodology

The KNOMAD methodology addresses the *process of developing* knowledge-based applications that have to cope with knowledge change. The following research questions are associated with the research objective of methodology development:

- How can knowledge change be accommodated during knowledge-based application development?
    - Which steps are required?

KNOMAD has been introduced as a methodology for the development of knowledge-based applications that can cope with changing knowledge. The critical aspect of knowledge change (and associated maintenance) is accounted for by the characterisation and analysis of knowledge change at the start of the KNOMAD process. Furthermore, inclusion of the KLC ontology within the modelling step of the KNOMAD methodology and development of a domain ontology in the organization step are specifically advised to ensure usability and

maintainability. Furthermore, KNOMAD adds an Analysis step to be able to assess knowledge-based application performance after completion of a project.

KNOMAD shares similarities with other KBS methodologies such as CommonKADS (Schreiber *et al.*, 1999) and MOKA (Stokes, 2001), such as justification for the business case, knowledge capture, knowledge organisation and delivery. It sets itself apart in the following ways:

- Explicit support for the identification of knowledge change in the Knowledge Capture & Identification of Knowledge Change step.
- Accounting for usability and maintainability aspects through the Organization and Modelling & Implementation steps.
- Explicit support for assessment of knowledge-based application performance.

Despite these differences, it must be stressed that these methodologies (and the associated models) are not mutually exclusive. Methodology steps and models can be used in conjunction with KNOMAD and the KLC ontology. For instance, the CommonKADS task and inference models can be used to develop the **Enterprise_Knowledge_Resource** and **Process** classes and the domain knowledge model can be used for the **Knowledge** class of the KLC ontology.

The KNOMAD steps have been substantiated to a limited degree in Section 3.3. In Chapters 4-6, case studies are presented that use the KNOMAD methodology as a guide for development. These case studies provide additional insight into the application of KNOMAD in multiple aircraft lifecycle phases.

3.5 Proposing a Case Study approach

Three case studies will be performed to validate the contributions to theory developed in Chapter 3. In each of the cases, a knowledge-based application is developed to perform an engineering task in which knowledge is liable to change. To position these applications, the following must be taken into consideration: they include a knowledge base, include capabilities for document and configuration management, feature (partial) automation of engineering tasks and feature analysis and/or optimisation capabilities. However, the applications have *not* been built using KBE systems and *do not include geometric handling capabilities*. Consequently, the developed solutions cannot be seen as KBE applications, but are simply labelled as *knowledge-based applications*.

To validate knowledge change for the engineering tasks associated with the specific cases, change will be characterised for each case study (Chapters 4-6). To validate the Knowledge Lifecycle Model, the maintenance case study (Chapter 6) will include a quantitative analysis of Airworthiness Directives to see whether the Knowledge Lifecycle Model's concepts can be applied to meaningfully quantify knowledge change in practice.

To facilitate knowledge change, the KNOMAD methodology and KLC ontology are used in the development of the knowledge-based applications. The applications have been developed to a proof-of-concept stage; though the knowledge lifecycle attributes have been implemented in the relevant models, the proof-of-concept status has meant that the applications have not been used structurally and for a longer period of time. As a result, the knowledge lifecycle concepts of state and action have not been actively used to quantify knowledge change relative to the developed knowledge-based applications.

Furthermore, the proof-of-concept status of the applications has had implications for the formality of the KLC ontology. Formal modelling of axioms, theorems and proofs thereof has not been required, leading to the selection of UML for development of the KLC ontology and consequently also for use in the case studies. As part of applying the KLC ontology, case-specific task and domain ontologies are developed and expressed in UML. These are subsequently implemented in a tool called Ardans Knowledge Maker (AKM). Consequently, the ontologies serve as the backbone of the knowledge base and knowledge-based application code maintained in AKM – an *ontology-based approach to knowledge lifecycle management* has been followed. It should be noted that for *all* case studies, the presented UML ontologies have been implemented in AKM.

AKM functions as single point of access to systematically store, access and manage the lifecycle of EKRs, knowledge and knowledge elements. It consists of a web-based interface on top of a knowledge base implemented in MySQL. Knowledge base elements are represented through AKM articles. AKM supports XPATH query language to identify and fill article fields by retrieving node information from the XML data that comes from MySQL. For instance, XPATH expressions can be used to let knowledge, knowledge element, process and process element models inherit common metadata and attributes. The interface allows users to set up an environment to directly access articles (e.g. EKRs) or users can employ a search environment to find specific articles using search spaces, context tags (metadata / class hierarchies) and search text.

Each of the case studies is structured in a similar way. The approach is as follows:

- **Case study context:** for each lifecycle stage, the context and specific engineering task for which a knowledge-based solution has to be developed are briefly introduced. The case study objective and requirements are explicitly stated.
- **Application of theory contributions to case study:**
    - Application of Knowledge Lifecycle model: the Knowledge Lifecycle model is used to identify and characterise knowledge

change for the case study engineering task. For the maintenance domain, the model is also applied to quantify knowledge change.
    - Application of KLC ontology: in preparation of application of the KLC ontology in solution development, the case study engineering task is analysed.
    - Application of KNOMAD: to prepare the solution development, application of the steps of KNOMAD is planned.
- **Solution development:** the KNOMAD methodology and the KLC ontology are used in the development of a knowledge-based application for the case study. The KNOMAD steps are adhered to in the following manner.
    - **Knowledge Capture & Identification of Change:** the objectives and scope of the case study are identified and the prerequisite knowledge is captured, presented and analysed for knowledge change;
    - **Normalization:** the captured knowledge is checked against pre-set criteria;
    - **Organisation:** a domain ontology is (partially) developed for the case study domain. The domain ontology is developed as an extension of the KLC ontology. The domain ontologies as represented here are developed to support the solution. As such, only those classes, attributes and relations that are relevant for the case study are implemented within the domain ontology. This prohibits the design and implementation of cumbersome, extremely large domain ontologies of which only a few concepts would be used;
    - **Modelling & Implementation:** the KLC ontology principles and classes are applied to the case, with specific attention to the development and implementation of a task ontology (through the Enterprise Knowledge Resource concept) for the engineering task at hand. The domain and task ontologies are subsequently implemented in AKM and the solution is built;
    - **Analysis & Delivery:** a proof-of-concept application is analysed; performance and functionality is validated relative to the case study objectives.
- **Discussion:** case study results are discussed in the context of the research objectives and contributions.

Note that two types of contribution and validation occur! Each case study can be seen as a contribution in its own right: a specific engineering problem is solved; an improved solution is implemented. As such, performance of the case study

solution is discussed in the **Analysis & Delivery** step to validate the case study contribution.

Furthermore, each case study acts as validation towards the overall research contributions. For each case study, the **Discussion of Results** section is used to discuss the case study in light of the contributions to theory. Furthermore, the case study results are synthesized in Section 7.1 as part of a wider view on the overall contribution with respect to theory. This leads in to the overall research conclusions, followed by a consideration on the limitations and recommendations associated with the performed research.

4 Design Case Study: Ply Stacking Sequence Optimization for Composite Wing Panels

This chapter describes the development of a knowledge-based application that automates blending of ply stacking sequences in the design of composite wing panels.

This case study belongs to a set of three case studies, with this case focusing on the design domain of the aircraft lifecycle. Together, the case studies will shed light on how the overall research objective can be achieved, with emphasis on the latter part of the objective: "Support consistent formalization, use and maintenance of changing knowledge within aircraft lifecycle phases *to improve domain-specific modelling, execution and control of engineering tasks*". The case studies also offer a practical perspective on the following research questions:

- How can knowledge change be accommodated during knowledge-based application development?
    - Which models are required and how do these models help to accommodate knowledge change?

The following section introduces the problem statement for this specific case study in the aircraft design domain. After this, the developed theory of Chapter 3 is applied to the case study. Subsequently, results are discussed in Section 4.3. A solution has been developed for the case study problem; development and implementation are discussed in detail. Validation of performance with respect to the case study objective(s) and requirements is briefly indicated in Section 4.3.5: Analysis & Delivery. The case study concludes with a discussion of the results within the context of the dissertation objectives and contributions to theory.

4.1 Case Study Context and Challenges

This case study research has been carried out in the context of the development of a new generation narrow-body civil aircraft, the Airbus A30X, pitched as an eventual successor to the A320 family of narrow-body aircraft. According to Airbus, A30X is to enter the market in the late 2020s (Airbus, 2011).

Although the market availability of the A30X aircraft is at least 15 years ahead from today, the complexity of an aircraft program forces research and technology development efforts to be started well in advance. Broadly speaking, the two main work streams to develop the A30X program are:

- **Conceptual design studies** to identify the optimum configuration of the aircraft to satisfy the forecasted market needs. In this work

stream, Airbus is evaluating novel aircraft configurations including forward-swept wings, rear-mounted turbofans and vertical tail planes among others (Aviation-Week, 2011).

- **Engineering capabilities** to effectively address technological challenges emerging from the new program. While the previous work stream focuses on design, this one aims to realize the necessary technology innovations that will achieve the market claims of the new aircraft (i.e. high fuel efficiency, low production cost and others). Examples in this direction include the development of new generation engines and the development of new design and manufacturing technologies to support the use of thermoplastic-based composites (Compositesworld, 2011). Tools and methods are concurrently developed to support the development of these engineering capabilities. These new technologies, tools and methods have a learning curve: to support engineers, fast and accurate access to up-to-date knowledge is necessary.

Contemporary industrial policy aims to encompass these two aerospace product development work streams through collaborative research projects. In the United Kingdom, an example of such a project is the Next Generation Composite Wing (NGCW) project (Northwest-Aerospace-Alliance, 2010), in which Airbus UK and EADS Innovation Works work with several major industrial and academic partners to research materials, technologies and tools to support the design and manufacturing of composite wing structures. As part of the NGCW work package, the Multi-Disciplinary Optimization of Wing (MDOW) research project has been initiated to research and develop design tools for the multidisciplinary analysis and optimization of wing structures. The work reported in this case study has been carried out as part of the MDOW project.

The focus of the case study is on the optimization of a composite wing cover conceptual design for ply continuity through the blending of stacking sequences. The optimization of composite airframe (part) design using stacking sequence blending is an area of research in its own right (Soremekun *et al.*, 2002; Gunawan *et al.*, 2003; IJsselmuiden *et al.*, 2009). It must be emphasized that the focus of this case study is on the application of the Knowledge Lifecycle Model (Section 3.1), KLC ontology (Section 3.2) and KNOMAD methodology (Section 3.3) to develop a knowledge-based solution that can cope with knowledge change. The technical details associated with the stacking sequence optimization routine embedded in this knowledge-based solution are not discussed. An example of a in-depth research project that addresses both stacking sequence optimization and the development of a knowledge-based engineering application for this problem is described in Cooper (2011).

The optimization of ply continuity in aircraft composite wing conceptual design is an example of addressing manufacturing considerations in an early stage of design. The industrial partner that participated in the case study used a grid representation in the conceptual design of a carbon fibre reinforced plastic (CFRP) wing. In each of the grid cells, the amount of carbon fibre plies and their orientation is specified, based on structural requirements (minimum thickness and load cases). This single-cell specification, known as “ply stacking sequence”, describes a particular sequence of composite layers, each of which has a specific fibre orientation (see Figure 4.1, where green layers denote a 0° fibre orientation, red and pink layers denote +45°/-45° fibre orientation, and blue denotes a 90° fibre orientation).

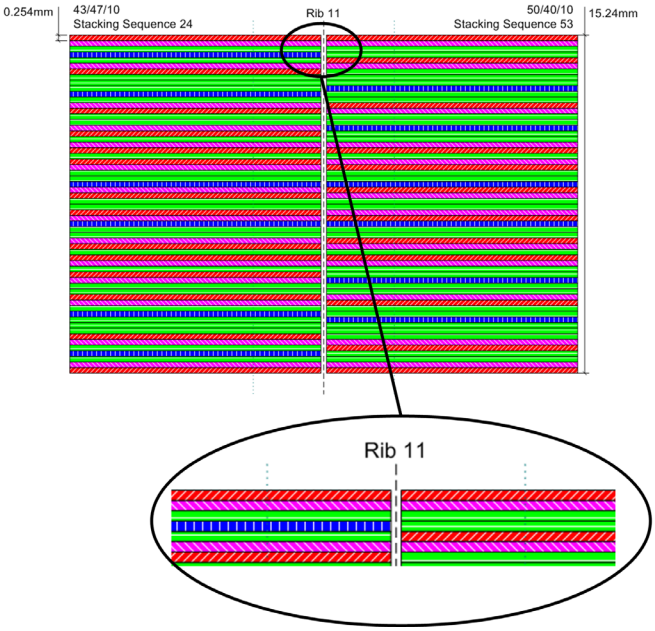

Figure 4.1: Cross-sectional view of ply stacking sequences for two adjacent grid cells

The out-take in Figure 4.1 shows that there can be mismatches between adjacent cell fibre orientations: in the out-take, the top three layers are consistent, but the remainder of the layers is not. For example, consider the fourth layer from the top. The left grid cell has a 90° fibre orientation (blue) and the right grid cell has a 0° fibre orientation (green): the orientations are discontinuous.

However, in composite manufacturing, material is not deposited in discrete cells but in continuous layers. The manufacturing of laminate plies is most cost-effective when continuous surfaces can be laid down, as machine start-up times, repositioning and material waste are kept to a minimum. Therefore,

discontinuities between individual cell stacking sequences must be kept to a minimum.

When manufacturing a product, discontinuities can be solved by introducing overlap and interleaving the adjacent layers (Figure 4.2). This figure shows that plies are extended over the rib area and 'stacked' on top of each other, which introduces additional thickness (and mass) at the rib area, and consequently a ramp gradient from the rib to the cell.

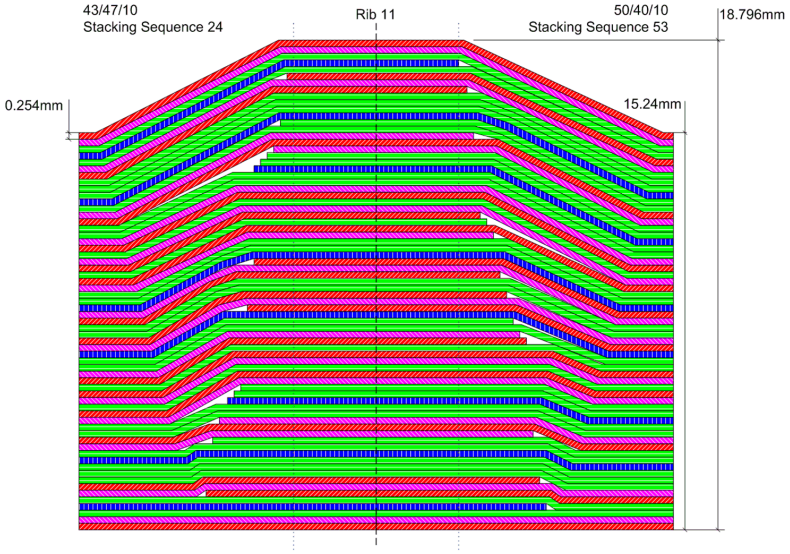

Figure 4.2: Interleaved plies across cell boundary

This ramp gradient must be kept within a specified limit depending on maximum tool deflection to ensure manufacturability. The result of adding interleaved plies at cell boundaries is shown in Figure 4.3.

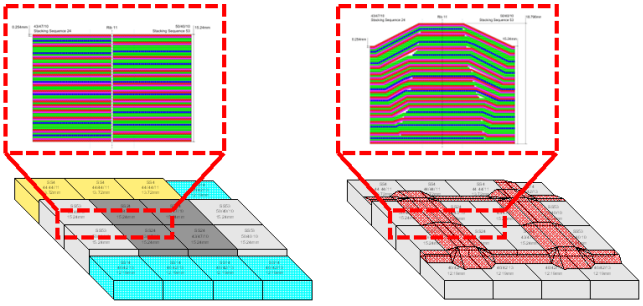

Figure 4.3: Adding manufacturing considerations into a structural view of the design

However, this solution adds mass. The full size of the problem becomes intuitively apparent when considering a set of adjacent stacking sequences for a complete panel such as the one in Figure 4.4. The wing cover skin panel shown in this Figure consists of many individual grid cells (as represented by the various coloured cells containing cell names), which bounds are formed by ribs and stringers.

Figure 4.4: Grid representation of a wing cover skin panel

Instead of introducing additional mass through overlap and interleaving during manufacturing, it is preferable to reconfigure adjacent stacking sequences *beforehand* such that ply continuity is optimal and mass addition is minimal, while respecting the structural design requirements (see also Section 4.3.1). In other words, if manufacturing considerations could be integrated into structural design and sizing, the material and therefore mass which is added later for manufacturing purposes can be reduced. The preferred solution for the ply continuity problem comes down to re-sequencing and optimizing a set of stacking sequences such that ply continuity is maximized and minimum addition of mass is achieved, while obeying structural design requirements.
Achieving a satisfactory design solution requires a high amount of manufacturing knowledge together with a high degree of automation in order to cope with hundreds of interfaces between cells and their stacking sequences. This problem can be tackled through a 'traditional' KBE approach where the knowledge is directly encoded into a software application. Such an approach would however disregard that many current aircraft design projects, in their switch towards the use of composite materials, are having to gain knowledge about design while designing: designers are “learning by doing” (Siemieniuch and Sinclair, 1999), and knowledge is subject to change.

The consolidated objective of the case study is to develop and implement a proof-of-concept knowledge-based solution for ply continuity optimization in composite wing panel design. The following requirements must be met:

1) The solution must be able to optimize a composite wing cover conceptual design for ply continuity through blending of stacking sequences;

2) The solution must give the possibility to trade off wing panel concepts on weight, cost and manufacturability;
3) The solution must be automated.

Validation with respect to these requirements is performed in Section 4.3.5: Analysis & Delivery.

The introduced problem is related to knowledge change and consequently to research challenges regarding knowledge usability and maintainability. The following issues will be taken into account in the case study:

- **Moving beyond black-box applications and ensuring transparency:** As mentioned before, a current drawback of many knowledge-based applications is that they are 'black-box': the knowledge contained in the these applications is difficult to access and inspect, and is often embedded in the application code (Verhagen *et al.*, 2012). To support knowledge maintenance, it is necessary to move beyond black-box processes and applications by supporting categorization, accessibility, traceability and subsequent sourcing of knowledge, which opens up the potential for knowledge reuse (Markus, 2001; Verhagen *et al.*, 2012). In order to allow users to inspect, use and maintain knowledge, it is necessary that the knowledge solution is transparent. I.e., it should be clear which knowledge is involved within a process, which further inputs are required, which steps are taken within a process and which outputs are generated.
- **Task orientation:** knowledge involves a 'capability for effective action'. The capability for action can be met by explicitly associating sets of knowledge with functional tasks, i.e. the optimization for ply continuity for this case study.
- **Expert / end user involvement:** End users must be able to identify, use, interact with and if necessary, maintain or update the relevant knowledge that they use in their daily work and specific context (Merali and Davies, 2001). For this case study, the design and manufacturing rules and constraints are of primary interest.

Through these considerations, the case study contributes to validation of the overall research contributions to theory. This is discussed in Section 4.4: Discussion of Results.

4.2 Application of Theory to Design Case Study

Before developing a solution, this section acts as an intermediate step by applying the developed theory to the case. First, the Knowledge Lifecycle Model is applied

to identify knowledge change for the ply continuity optimization task. This task is subsequently analysed in support of further application of the KLC ontology to solution development in the Results section. Finally, it will be shown how the KNOMAD steps will be applied to this case to guide the subsequent solution development in Section 4.3: Results.

4.2.1 Application of Knowledge Lifecycle Model: Identifying Knowledge Change

For this Design for Manufacturing (DFM) problem, some of the required design knowledge is not subject to change. For instance, the design rule with respect to tension or compression loads (see Section 4.3.1) which states that the percentage of 0° fibres can be increased to improve tension and compression laminate properties will remain valid as the basic physics underlying this rule do not change.

However, there are a number of examples in which changes in knowledge and associated specifications would have a dramatic impact on the optimization of design solutions. A number of qualitative examples are given below:

- **Design methods:** maintaining ply continuity between adjacent cells (or panels) of laminates, also known as blending, is a research area in its own right. Various methods and improvements have been proposed, as analysed by Liu *et al.* (2010); in the period of 2000-2009, at least 16 papers (including high-impact journal publications, e.g. Liu *et al.* (2000); Kristinsdottir *et al.* (2001); Soremekun *et al.* (2002); Liu and Haftka (2004); IJsselmuiden *et al.* (2009)) have been written to report on new, improved or successfully applied methods for blended design of composite structures. Or, to use Knowledge Lifecycle model terminology, design methods have been created, updated and maintained over the years. A search in Elsevier's Scopus search engine on 'composite blending optimization' adds a further eight recent journal papers reporting on advances in this field (Gillet *et al.*, 2010; Liu *et al.*, 2010; Bruyneel, 2011; Jin *et al.*, 2011; Liu *et al.*, 2011; Bruyneel *et al.*, 2012; Panesar and Weaver, 2012; Zein *et al.*, 2012). This shows that existing methods for blending are subject to change, which indicates that design knowledge for this specific area is subject to change. Ideally, any design process that is encapsulated within a knowledge-based solution should be able to change to reflect current best practice and/or extend applicability.
- **Tooling specifications:** the intended layup process to produce the conceptual wing designs uses Automated Tape Laying (ATL) or Automated Fibre Placement (AFP) techniques. If the specifications of the ATL / AFP

machines are updated, the associated optimization constraints can be relaxed. Two examples constituting data and information change are:

- Ramp gradients: ramps can occur between cells (representing stringer and rib bays), dependent on the required thicknesses to satisfy structural requirements. The designed ramp geometry should not exceed roller deflection, such that the roller remains in contact with the material when performing lay-up. If ATL/AFP maximum roller deflection is increased, the associated ramp gradient requirement can be relaxed.
- Minimum course (cut) lengths: ATL and AFP machines cannot cut tape shorter than a specified minimum course (or cut) length. This has the potential to cause production issues – see Figure 4-5. This Figure shows a scenario in which the minimum cut length exceeds the required tape lengths at the boundaries of the rectangular part. In practice, this will result in additional ply material at those boundaries, either leading to increased mass and (potential) overlaps with neighbouring parts, or to additional machining and material waste involved in cutting off the excess material at the edges.

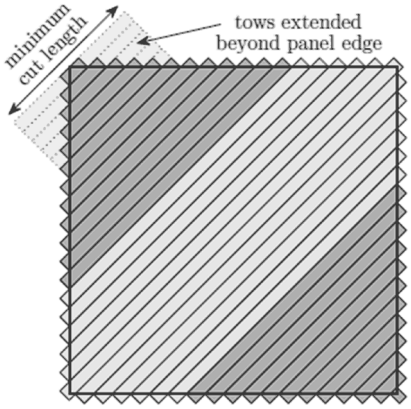

Figure 4.5: Possible production issues arising from minimum course length (Blom, 2010)

If the minimum course length is decreased, the manufacturing of smaller, more varied and more precise layer geometry is feasible.

- **Material specifications:** changes with respect to material behaviour (e.g. 'update' actions that introduce revised knowledge regarding lay-up or curing behaviour, or 'maintain' actions that extend the applicability of a material) can influence design and manufacturing requirements. Consider the two following examples:

- Maximum ply drop: a current constraint is that no more than 4 plies can be laid down sequentially without introducing a covering ply. This prevents delamination at the ply edges. Improved material specifications can increase the number of plies for sequential ply drops.
- Stringer ramping: a current constraint dictates that the rate of change of thickness of the laminate beneath stringers is restricted to reduce defects forming in the stringer blade. Improved material specifications can reduce this restriction.

The preceding examples qualitatively show that knowledge related to the ply optimization problem can indeed be subject to change and can loosely be characterised using Knowledge Lifecycle model concepts. However, given the proof-of-concept status of the developed solution (see Section 4.3), it has proven impossible to use the solution to quantify changes in the underlying knowledge.

4.2.2 Application of KLC Ontology: Task Analysis

The case study objective is to achieve ply continuity optimization in composite wing panel design. This design task incorporates design and manufacturing constraints that are applied to the wing panel structure in order to optimize it for weight, cost and manufacturability. In this section, the task is analysed into more depth in preparation of the actual use of the KLC ontology in solution development (see Section 4.3).

The existing process for solving the problem at hand is contained within a solution known as mPDA (manufacturable Ply Design and Analysis). mPDA is a solution developed in-house at the industrial partner. It has been implemented using Microsoft Excel and VBA. It requires the input of a specification file from the structural engineering department that specifies the wing cover grid, grid cell thicknesses and associated stacking sequences (a so-called fishtail plot). The exact content of the existing process itself is considered proprietary and cannot be fully reproduced here: only the general activities of the task are analysed below.

The top-level task is modelled in Figure 4.6 using an IDEF0 representation. The figure highlights the central task: optimize wing cover for ply continuity (A-0). This task requires input from the structures department in the form of a ply specification file. This input is processed by the ply optimizer tool (mPDA; see also Section 4.3.1) to generate an optimized ply specification. The ply optimizer takes into account various design and manufacturing constraints.

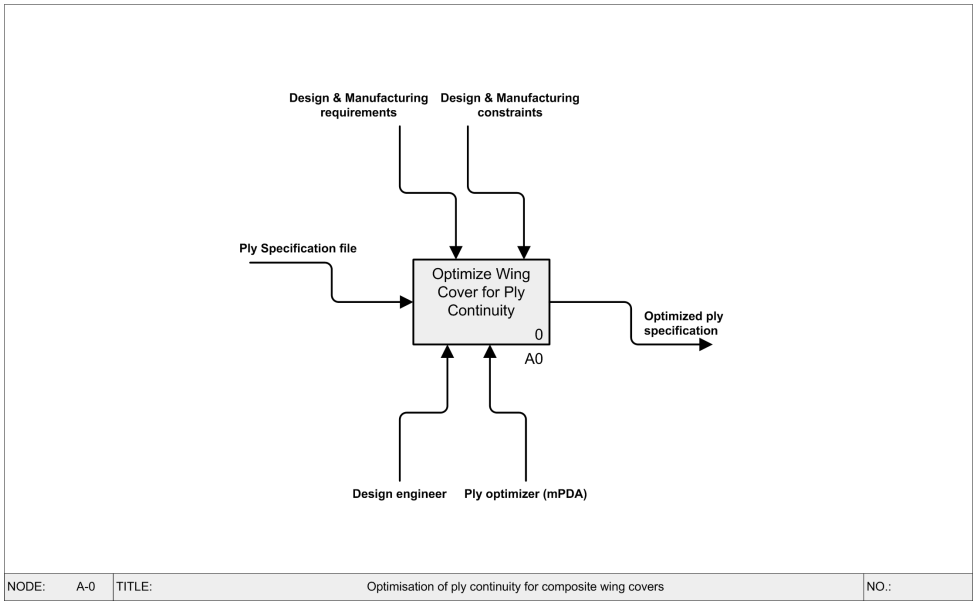

Figure 4.6: IDEF0 A-0 diagram for composite wing cover optimization task

The top-level task can be split up into four subtasks (see Figure 4.7). The preparation task (A1) takes panel sizing information (minimum required thickness and thickness law per grid cell) from the ply specification file to generate a catalogue of stacking sequences. This catalogue is used in the processing task (A2), which uses the ply specification file as an input for the generation of ply fishtail plots – a set of fishtail plots is generated to indicate grid coverage per ply, instead of one specification file containing thickness and/or stacking sequence per grid cell. In effect, the specification file is decomposed into a set of (virtual) fishtail plots indicating the grid coverage per ply layer. The optimization task (A3) applies the design and manufacturing requirements and constraints to the ply fishtail plots to configure optimized ply fishtail plots. Finally, the post-processing task (A4) uses these plots to put together an optimized ply specification file, where the wing cover has been optimized for ply continuity.

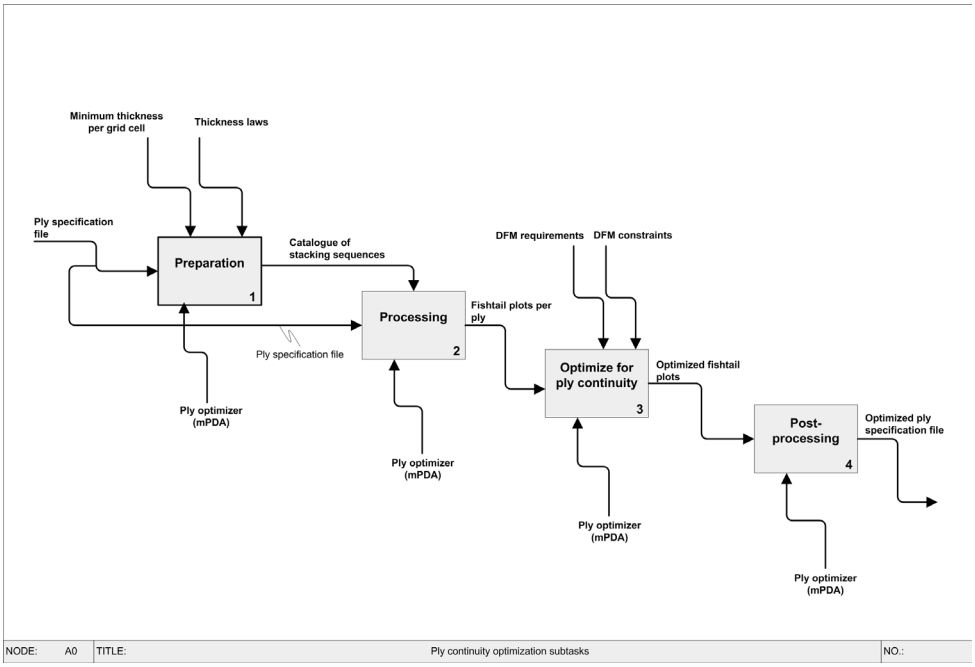

Figure 4.7: IDEF0 A0 diagram for optimization subtasks

4.2.3 Application of KNOMAD: Solution Approach

The KNOMAD methodology as discussed in Section 3.3 is adopted to develop a knowledge-based system for the case study problem. The KNOMAD steps relative to this case study are shown in Figure 4.8. This figure shows the main KNOMAD steps (Knowledge Capture & Identification of Knowledge Change; Normalisation; Organisation; Modelling & Implementation; Analysis & Delivery) with the associated activities that are required for this particular case study.

In the first step (Knowledge Capture & Identification of Knowledge Change), the justification for and scope of the knowledge-based system is established, followed by capture of the knowledge and process elements. The design and manufacturing constraints as well as the inputs to the problem are of particular interest. As knowledge change for this case study has already been considered in Section 4.2.1, this activity is not repeated. For the second step (Normalisation), the focus is on checking data quality and establishing input and output formats. The third step (Organisation) considers development of a domain ontology that holds the relevant concepts and relationships for this particular case study. It is split up into three parts: generation of product, process and resource class diagrams. The fourth step (Modelling & Implementation) concerns the development of models (in the Modelling sub-step), architecture and solution (in the Implementation step). The developed task and domain ontologies are

implemented in AKM to support the developed solution, making the solution ontology-based. Finally, the Analysis and Delivery steps are combined into one: performance of the solution is assessed relative to the requirements, and the costs and benefits of the solution are explored.

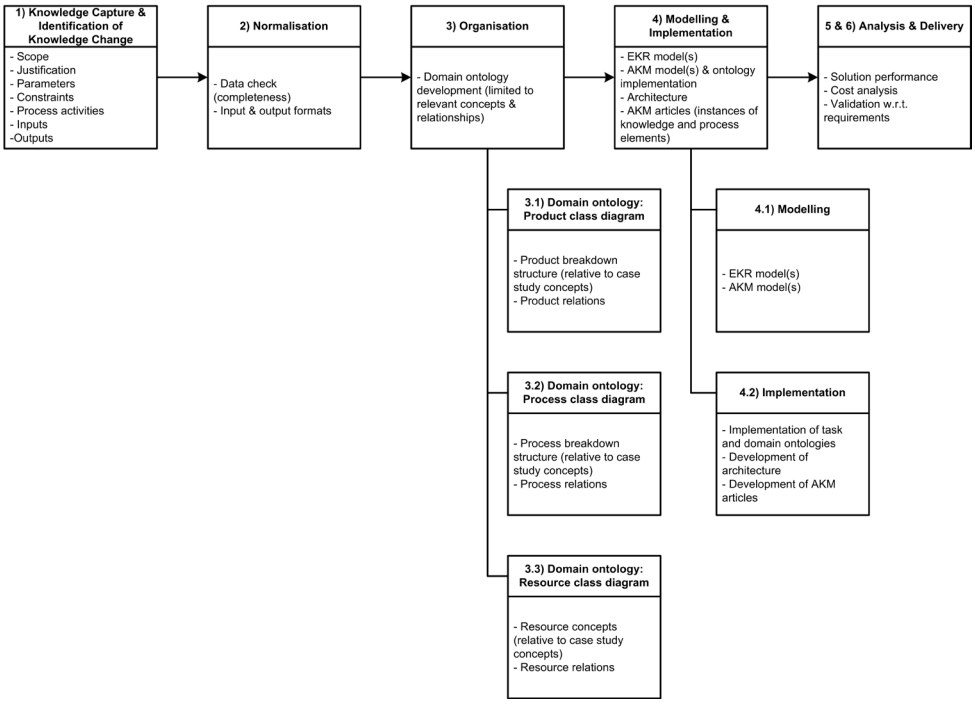

Figure 4.8: Application of KNOMAD to design case study – flow chart

4.3 Results

The next section describes the development of a knowledge-based application for the ply continuity optimization problem. The proof-of-concept solution can cope with knowledge change and addresses the associated issues of knowledge usability and maintainability. The development of the solution is based upon application of the revised KNOMAD methodology. Furthermore, the principles and concepts of the KLC ontology are used. The following sections are compliant with the KNOMAD steps.

4.3.1 Knowledge Capture & Identification of Knowledge Change

The composite wing cover ply continuity optimization task has been identified in Section 4.1 and further fleshed out in Section 4.2. With respect to justification of the business opportunity, the current process requires significant manual

intervention. Furthermore, part of the current solution is automated but this has been implemented in Excel and VBA macros, which leads to time-intensive optimization. Most significantly, the current solution is black-box: the knowledge used to optimize ply stacking sequences for continuity is not visible at all, but embedded in the mPDA application. It was decided to resolve these issues by developing a proof-of-concept knowledge-based solution.

The first step in the development of the knowledge-based solution is to capture the required knowledge elements. This knowledge can be extracted by considering two elements: the current process for solving the problem, and the set of inputs necessary for executing the process.

The existing process for solving the problem at hand is contained within a pre-existing solution known as mPDA (manufacturable Ply Design and Analysis); it has been discussed in Section 4.2.2, together with the high-level task and subtasks that are associated with solving the problem.

The second vital element of knowledge considers the set of inputs necessary for executing the process. In the current context, this pertains to the set of design and manufacturing constraints that must be complied with when solving the stacking sequence optimization problem. This set has been elicited from composite manufacturing experts. In total, 30 design and manufacturing constraints have been made explicit; a simplified overview of 30 of these constraints is given in Table 4.1.

Besides the constraints, the process requires a specification file to run. This specification file comes from the structural design department and contains a grid plot of the designed wing cover, with specific stacking sequences for each grid cell.

Knowledge change for this case study has been described in Section 4.2.1 and is not further analysed here.

Table 4.1: Captured design and manufacturing constraints

Constraint name	Description
Ply step thickness	Total thickness must be divisible by ply thickness
Practical ply quantities	The quantity of plies for each orientation percentage must be an integer
Paired diagonal plies	The number of 45- and 135-degree orientated plies should be equal
Symmetrical paired plies	The quantity of 45- and 135-degree orientated plies should be an even number, or the odd pair must be moved to the neutral axis
Tension/compression optimization	Improve the laminate properties for tension or compression; increase the percentage of 0-degree fibers
Shear load optimization	Optimize thickness law or stacking sequence to suit a shear load case; ncrease the number (percentage) of diagonal (45/135) plies to improve shear load performance
Multi-load optimization	Optimization variables for multi-load applications; equalize the percentage of each fiber orientation
Buckling optimization	Optimization for improved buckling resistance; move 0-degree plies away from the neutral axis to improve buckling resistance
Reduce inter-laminar shear	Optimize the stacking sequence to reduce inter-laminar shear; re-sequence the stacking sequence so that 0-degree are not next to 90-degree plies
Laminate hole preparation	Areas of the laminate which will have holes drilled through need to be prepared for bearing load bypass; typical load case requires a minimum of 40% diagonal plies, shear strength load cases need a minimum of 50% diagonals
Pivotal sequence	If the total number of plies is an odd number, then the sequence should pivot ($) about the last ply in the list (which is in a central position).
Symmetrical sequence	If the total number of plies is even, then it should be symmetric (S) about the last ply in the list (which is in a central position in the actual sequence).
First ply coverage	The 'Surface plies' (a combination of one 45-degree ply and one 135-degree ply) must be continuous over the whole surface.
First ply orientation	The 'Surface plies' (a combination of one 45-degree ply and one 135-degree ply) must be either of 45- or 135-degree orientation.
Diagonal ply order	The 45- and 135-degree orientated plies must be paired.
Diagonal ply sequence	The 45 and 135 orientation plies should be sequenced in a consistent order (either as 45 followed by 135 in all sequences OR 135 followed by 45 in all sequences).
Consecutive ply grouping	There should not be more than 3 consecutive plies of the same orientation.
Consecutive diagonal ply pairing	Consecutive pairs of 45- and 135-degree orientation plies should be avoided.
Virtual ply sequence	The Stacking Sequence plies should be aligned with the Virtual Ply sequence.
Minimum ATL diagonal cut	If the stringer pitch is less than approx. 70% of the tape width, then the 45 and 13-degree plies must span at least 2 stringer bays.
ATL minimum course length	Ply width or length must not be less than 30 mm.
ATL angle cut	Ply boundary cannot be less than 8 degrees parallel to the tape orientation.
AFP minimum course length	Ply width or length must not be less than 80 mm.
Stringer clearance	Flat clearance must be left for all stringer foot locations.
Rib clearance	Flat clearance must be left for all rib foot locations.
Ramp geometry	Ramp geometry should not exceed roller deflection.
Ply drop-off pattern	Ply drop off scheme should preferentially be 'sock' or 'diamond'.
Stringer ramping	Rate of change of thickness is restricted to reduce defects forming in stringer blade.
Spar ramping	Wing skin must not ramp greater than 1/40 where it interfaces with a spar.
Maximum ply drop	No more than 4 plies can be dropped sequentially without introducing a covering ply.

4.3.2 Normalization

To ensure knowledge quality and compliance to a standard, measures have been performed on the captured knowledge:

- **Traceability & Ownership:** the captured knowledge has been recorded using a pre-defined format that has been designed and agreed upon during the

research process. The format is implemented in a knowledge management tool (Ardans Knowledge Maker, or AKM) and has two main components. First, an informal representation of a constraint is recorded; it consists of a natural language description explaining its applicability and properties and is accompanied by any relevant illustrations. It also includes metadata (data about data) such as ownership, authorship and date of creation. Secondly, a formal description of the constraint includes software code that can be accessed by the mPDA application. An example of a single manufacturing constraint is illustrated in Figure 4.9.

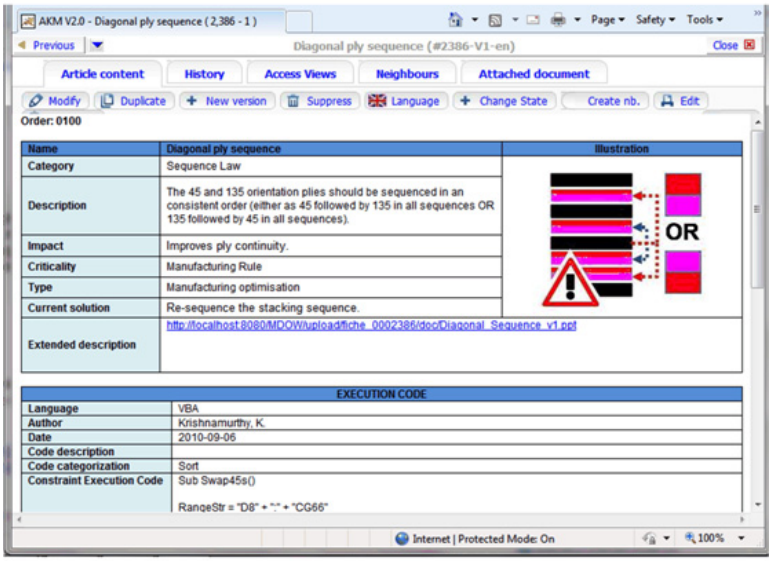

Figure 4.9: Example of manufacturing constraint stored in AKM

- **Accuracy & reliability:** the pre-existing process (mPDA) has been used as a baseline for construction of a knowledge-based application. The required design parameters (input, process and output) have been identified and captured. The resulting standardized formats are further explained in Section 4.3.4: Modelling and Implementation.

4.3.3 Organisation

The next step is to provide a knowledge structure that can be used to store the captured knowledge and can serve as the semantic backbone for the knowledge-based application. To achieve this, it is necessary to construct a domain- and case study-specific set of concepts and relationships: a domain ontology. For each of the case study domains, the high-level concepts and relationships of the KLC ontology (as given in Section 3.2) have been extended into domain-specific class

hierarchies. Based on the shared inheritance from the KLC ontology and the resulting use of the same high-level classes, many concepts and relations in the design, manufacturing and maintenance domain ontologies (as shown in Sections 5.3.3and 6.3.3) are the same. This consistency is desirable from a through-life perspective.

In this section, relevant excerpts of the domain-specific class hierarchies are given to explain how the design domain ontology is composed. Furthermore, these excerpts are specifically geared towards the classes, attributes and relationships that are necessary to develop a proof-of-concept solution for this case study. The domain ontology development is not exhaustive and can be extended using considerably more detailed concept representations, but it is considered sufficiently complete for the purposes of this case study, i.e., for use in annotation of the solution (elements).

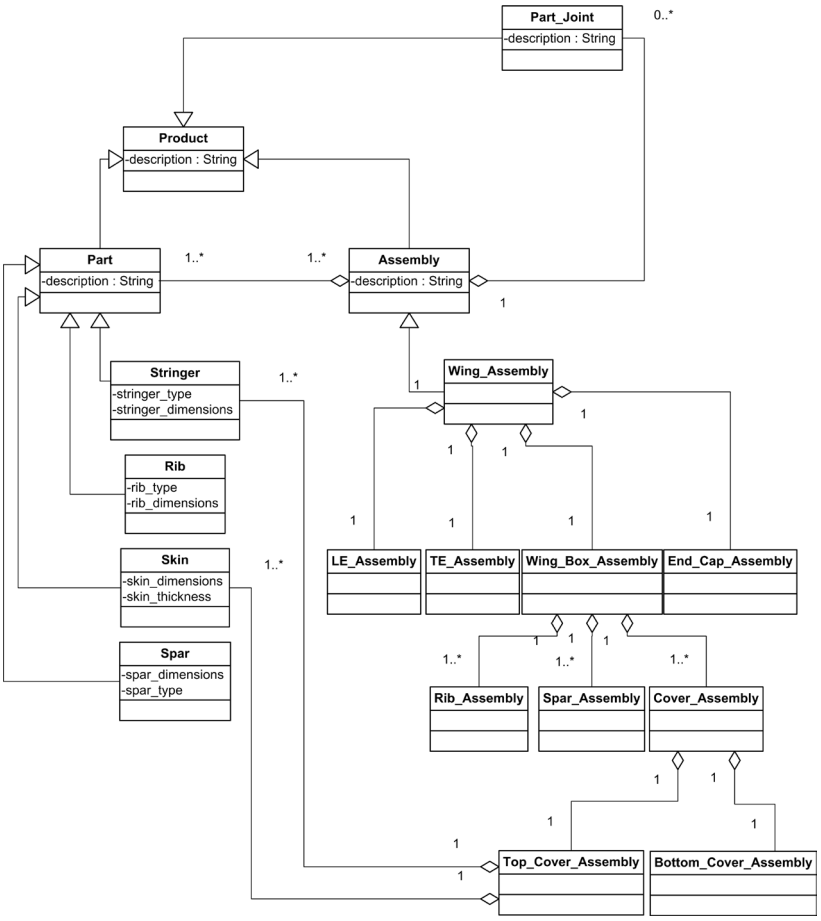

Figure 4.10: Domain-specific hierarchy for Product **class**

First of all, the domain-specific class hierarchy for the **Product** class is represented in Figure 4.10. The **Product** class hierarchy extends across a variety of composite products. For this use case, only composite wing covers are considered. However, though blending optimization is often considered for composite wings, the ontology must accommodate a suitable range of composites to which blending could be applied and must represent the context in which wing covers are designed. The **Assembly** and **Part** classes each contain several subclasses, including the wing box assembly, which consists of the cover, spar and rib assemblies. These assemblies contain one or more parts, which can be skin, spars, ribs or stringers. Figure 4.10 contains one example of assembly-part relations: the **Top_Cover_Assembly** class is composed of **Skin** and **Stringer** parts. Other assembly-part relations can be modelled, but to maintain figure clarity these are not included. The represented class hierarchies are not exhaustive and can be extended considerably, but are sufficiently complete for the purposes of this case study, i.e., for use in annotation using the PPR paradigm (see Section 4.3.4).

A number of example attributes have been added to the classes. For example, the **Stringer** class has an attribute stringer_type which can be used to express the general type of stringer, e.g. Z-stringer, L-stringer, T-stringer, Ω (omega)-stringer. Similar to the class hierarchies, the class attributes are not to be considered as complete, but rather as a representation of the most important attributes – many attributes have been omitted from the representation in Figure 4.10.

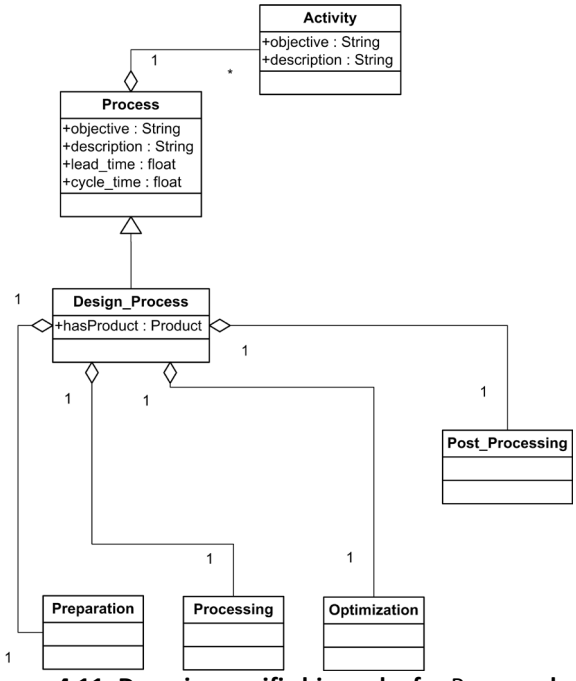

Figure 4.11: Domain-specific hierarchy for Process **class**

The domain-specific class hierarchy for the **Process** class is represented in Figure 4.11. For this domain-specific hierarchy, the focus is on **Process** subclasses that can be used for structuring and annotating domain knowledge. The task activities analysed in Figure 4.7 are used as subclasses of the **Process** class: **Preparation**, **Processing**, **Optimization** and **Post_Processing** have been modelled. These classes are fairly generic and can be used for many design tasks that incorporate optimization. The **Activity** hierarchy can be further extended to include the activities that make up the **Preparation**, **Processing**, **Optimization** and **Post_Processing** classes, but given the confidentiality of these activities (as indicated in 4.3.1), this is not shown here.

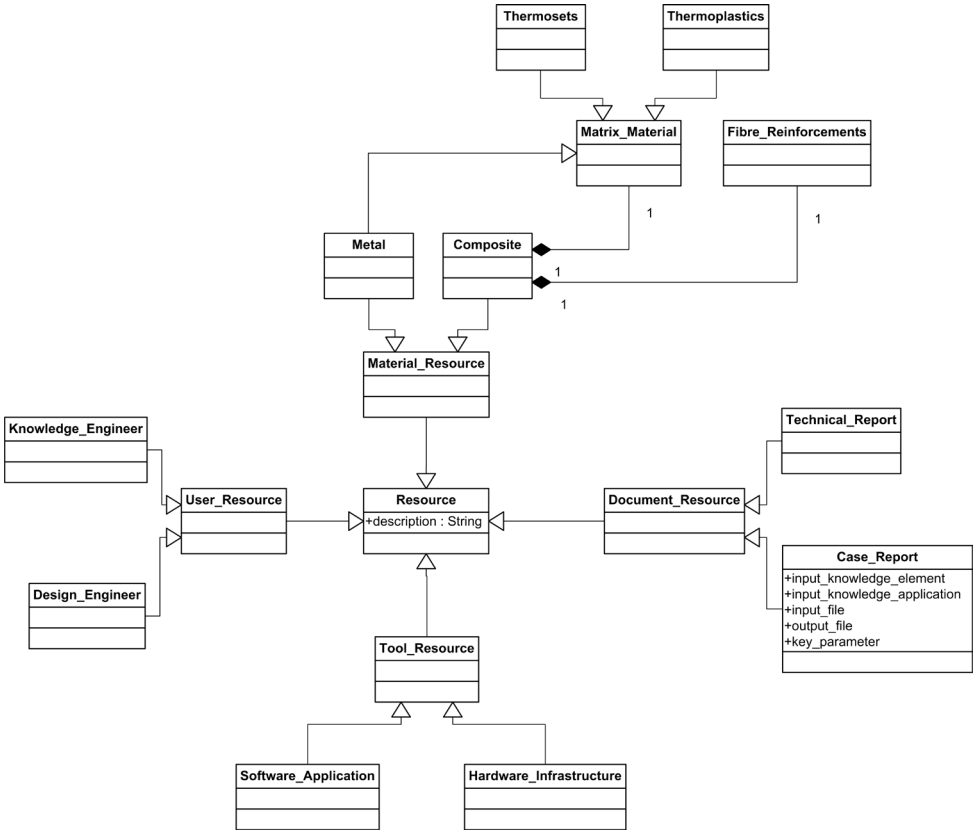

Figure 4.12: Domain-specific hierarchy for Resource **class**

The domain-specific class hierarchy for the **Resource** class is represented in Figure 4.12. The focus is on subclasses that can be used to structure and annotate domain knowledge. The resource classes include **Material_Resource**, **Tool_Resource**, **User_Resource** and **Document_Resource**. For each of these

classes, a few examples of subclasses are given to better illustrate the class hierarchy content. Instances of these subclasses can be related to the task analysis performed in Section 4.2.2 (Figure 4.7): for instance, mPDA is an instance of **Software_Application**, whereas DFM requirements and constraints, thickness laws and ply specification file(s) are examples of **Document_Resource** or even **Technical_Report**.

In order to keep the preceding domain class hierarchy figures clear, the relationships between classes and subclasses of the **Product**, **Process** and **Resource** trees, e.g. between **Part** and **Material_Resource**, have been omitted from the figures. The main relationships are given in Table 4.2, which can be seen as an addition to the high-level relationships identified in Table 3.7, Section 3.2.3.3. The *is-a* relationships have not been included into this overview, but these are given in the Figures using the broad-headed arrow UML format. Through inheritance, these *is-a* relationships allow for subclasses to inherit relations.

Table 4.2: Relationships in the design domain ontology

Class 1	Class 2	Relation (name)	Relation (type)
Part	**Material_Resource**	*hasResource*	Aggregation
Part_Joint	**Material_Resource**	*hasResource*	Aggregation
Part	**Document_Resource**	*hasResource*	Aggregation
Process	**Document_Resource**	*hasResource*	Aggregation
Design_Process	**Document_Resource**	*hasResource*	Aggregation
Design_Process	**Tool_Resource**	*hasResource*	Aggregation
Enterprise_Knowledge_Resource	**Tool_Resource**	*hasResource*	Aggregation
Enterprise_Knowledge_Resource	**User_Resource**	*hasResource*	Aggregation
Composite	**Matrix_Material**	*contains*	Composition
Composite	**Fibre_Reinforcements**	*contains*	Composition

The domain ontology as presented here has been used to structure the captured knowledge and will be used in the subsequent step to annotate (elements of) the knowledge-based application. This is further explained in the following Section.

4.3.4 Modelling & Implementation

This step consists of two highly related activities: modeling of an Enterprise Knowledge Resource for ply continuity optimization and implementation of the models into a functioning solution.

4.3.4.1 Solution Development: EKR Modeling

The first step in the development of a solution for the ply continuity optimization problem is to adopt the Enterprise Knowledge Resource approach, the

cornerstone of the KLC ontology. Based on this approach, models are required to represent and store:

1) A set of knowledge articles containing design and manufacturing constraints relevant to the design of a manufacturing-compliant ply stacking sequence. These constraints have been presented in Section 4.3.1.
2) A process model that models the activities (process elements) of the design and analysis process and combines knowledge articles, elements and code of the mPDA tool, and an input specification file of the wing cover to automatically execute the optimization problem.
3) A set of case reports storing the history of ply stacking sequence optimization results, as well as keeping traceability between the results obtained, the knowledge used to derive them and the inputs of the process.

Using the preceding considerations, an EKR class diagram has been modelled for this specific case study and associated task. The UML class diagram is shown in Figure 4.13.

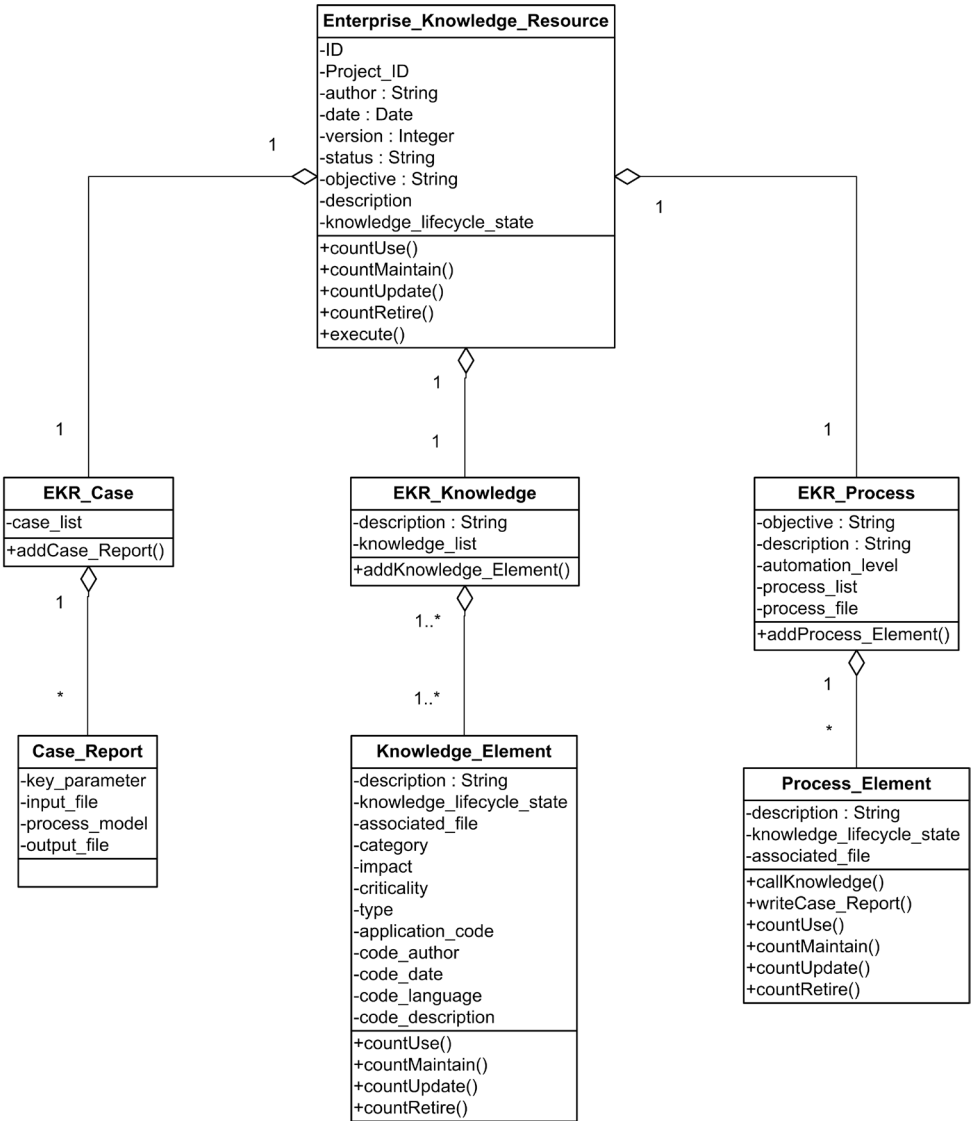

Figure 4.13: EKR class diagram (UML) for design case study

4.3.4.2 Solution Development: EKR Implementation

An architecture for the implementation, use and maintenance of EKRS for the design domain has been developed, with AKM as a major contributor. Two main architectural elements are used to deploy an EKR in practice:

- **EKR Environment for Learning by Doing (eLBD):** The environment for learning by doing (eLBD) is a web solution aimed at supporting end users. eLBD is based on AKM. The domain ontology as introduced before and

specific AKM models for the representation of knowledge and process elements (see further on in this section) have been implemented within AKM to enable the construction of EKRs which package the process and knowledge elements and the cases. The role of eLBD is not to store concept data but the collective thought behind the data (assumptions, constraints, rules, procedures and tools).

- **Executable environment for Learning by Doing (xLBD):** The executable environment for learning by doing (xLBD) is a solution to enable the remote execution of EKRs through a web service approach. xLBD uses several software applications and languages (Apache Tomcat web server, Java, AKM web services and Phoenix Integration Model Center®) to deploy the EKRs as web services. Users can access and execute the software remotely, so they do not require a dedicated installation of software on their desktops. Once the user is in the system, he or she is given an overview of EKRs that are available for use. This overview is based upon the user's security credentials, function and organizational position: for instance, a wing box designer for a specific aircraft programme will only be able to access EKRs that are related to the design and analysis of the wing box for that specific programme. Each EKR stands for a single design or analysis task that can be executed by the user. Depending on the task, the EKR can be executed automatically (i.e. the user presses a 'run' button and there is no subsequent user intervention; all software is run automatically in the correct order, and relevant inputs and outputs are passed on between process stages and eventually presented to the user), or the user is involved in the execution of the EKR (e.g. by selecting the right input files for a specific analysis task). Given the proper security permissions, the user can also inspect, retrieve and manage the knowledge contained in the EKR(s). Every time an EKR is executed, a case report is automatically populated. This report (based on the **Case_Report** class) registers the inputs supplied, the outputs generated and the knowledge used in the case.

The knowledge framework, with the eLBD and xLBD environments at its core, is shown in Figure 4.14. The eLBD environment allows users to access EKRs and their constituent elements, whereas the xLBD environment allows (remote) execution of an EKR.

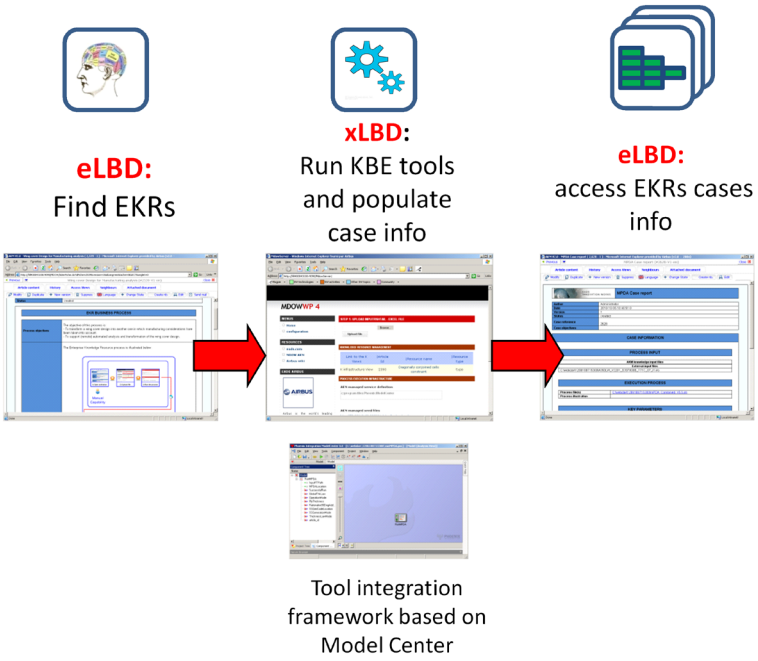

Figure 4.14: Knowledge framework containing the eLBD and xLBD elements (Bermell-Garcia *et al.*, 2012)

To implement the knowledge framework, models for the knowledge and process elements have been developed based upon the presented UML class diagram (Figure 4.13) and have been introduced in the Ardans Knowledge Maker (AKM) tool. The models are presented below.

First of all, a generic model for the representation of EKRs has been developed. This model is given in Figure 4.15.

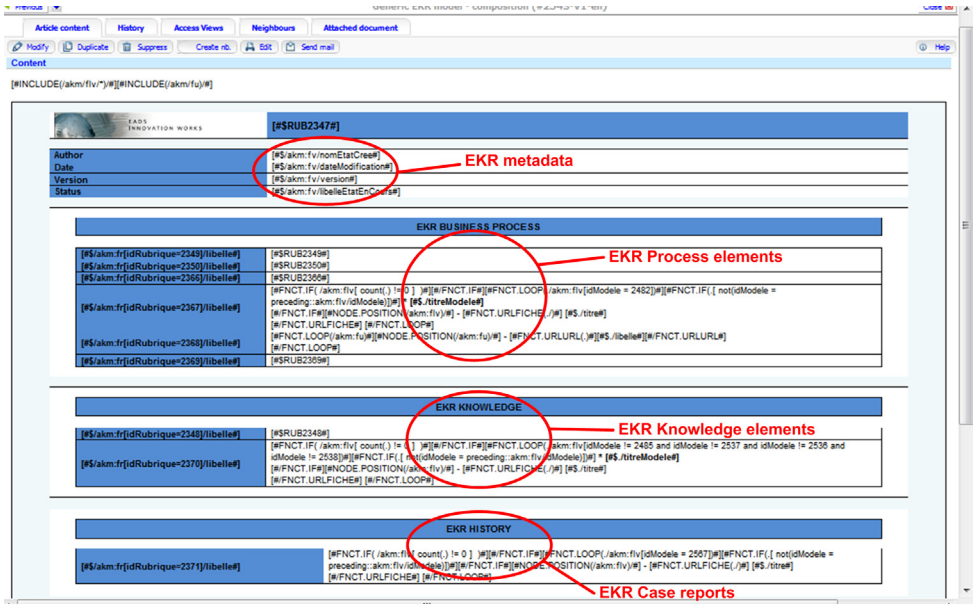

Figure 4.15: AKM model for Enterprise_Knowledge_Resource **class**

The EKR model lists metadata regarding authorship, date, version and status. The EKR model represents the three main elements of an EKR: **EKR_Knowledge**, **EKR_Process** and **EKR_Case**. The latter two classes are represented into the EKR model directly, instead of having separate models. Attributes such as objective, description and level of automation of the process model are represented. Furthermore, the EKR model part concerned with the process model links towards a process file: this file is used in the implementation to express the process activities and automate the execution of the process. Furthermore, the EKR model gives a list of cases: outputs of performing the EKR task. The individual cases are represented using their own model.

Two models have been used to represent knowledge elements for this case study. The first model reflects the **EKR_Knowledge** class of the EKR UML model and acts as a container for the set of individual knowledge elements. The **Knowledge_Element** class of the EKR UML model has been given a separate model. This has two main parts: first, an informal description of the knowledge element is given. This contains attributes for category, (extended) description, impact, criticality and type of the knowledge element. The second main part is the formal representation of the knowledge element: here, code metadata and actual code can be represented into the knowledge element.

Finally, a model has been created to represent the output of a design task. It is used for instantiation of the individual case reports. Its main elements are process input, execution process, key parameters and process output fields. The process input contains a list of the used knowledge elements and any

supplementary input files. The execution process field contains an overview of the used process elements. The key parameters field lists the main parameters used while executing the process. The process output field contains or links towards the output files that were generated after executing the process.

Using these models, a single EKR for the ply continuity optimization task has actually been implemented.

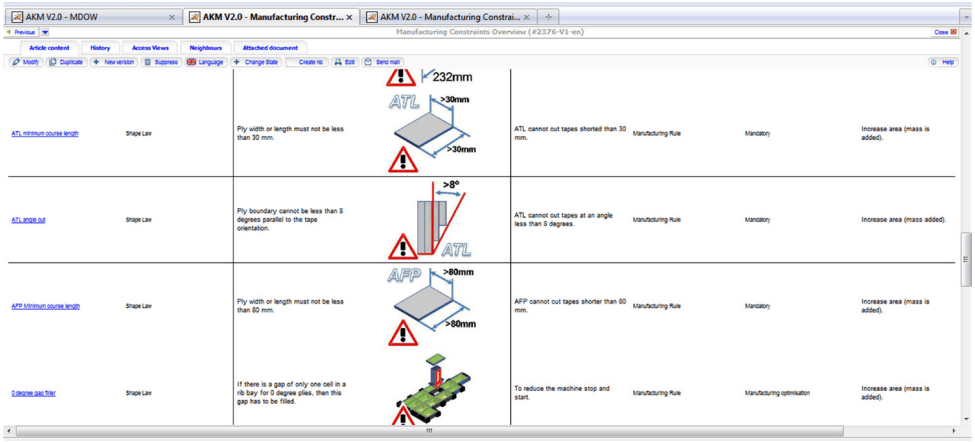

Figure 4.16: Partial overview of implemented design and manufacturing constraints (Bermell-Garcia *et al.*, 2012)

Figure 4.16 shows a partial overview of the implemented design and manufacturing constraints. For the individual knowledge elements (i.e. design and manufacturing constraints regarding ply continuity optimization), the AKM model supports informal and formal representation. Both representations of the constraint are stored in a single knowledge article associated to the EKR. An example of a single manufacturing constraint for eLBD users has been shown in Figure 4.9.

Using a combination of the specification file, the applied constraints and the knowledge-based application (mPDA), the blending optimization is performed. In the implemented version of the architecture, the process is modelled within Phoenix Model Center and uses mPDA as its main element to perform the blending optimization. The process is executed through the xLBD environment; a local or remote user can select the relevant input and constraints to apply for the stacking sequence blending and optimization. The user can then execute the process, which runs and completes automatically using the process model. The result is a composite wing cover design that takes into account design and manufacturing constraints. The access to this process is managed by the eLBD environment which points the user to the stacking sequence blending

optimization EKR, and consequently the related knowledge and outputs (in the form of case reports) can be inspected.

As introduced in Section 4.3.1, mPDA needs two main inputs to run: firstly, it takes a design specification file where the wing cover design is represented as a set of adjacent discrete cells, each having a specific stacking sequence. After importing this data, the different modules of mPDA place requests to the eLBD knowledge repository to retrieve the code of relevant design and manufacturing constraints to be evaluated. This is achieved at runtime by the use of AKM's data retrieval web service.

The final part of the implemented EKR for ply continuity optimization is embodied by the case reports. These reports store the optimized process results, as well as the used inputs, knowledge elements and process file. The case reports have been implemented in AKM and are stored as separate articles. When an EKR is run multiple times, for instance with multiple sets of different inputs, the results are gathered in a set of case reports that is listed under the **EKR_Case** element of an EKR. This enables the subsequent inspection of analysis results, but also opens up the opportunity to further analyze the results themselves. Case metadata is also automatically assigned, which enables consistent categorization and easier search and retrieval of historical analysis results.

As such, AKM is the central point for initiating analysis (accessing eLBD and initiating mPDA through the relevant EKR), inspecting the underlying knowledge (constraints), and inspecting and /or analyzing the results (case reports). The relations between the case reports and the associated content (design inputs; design and manufacturing constraints knowledge) have been illustrated in Figure 4.17, which shows some generic wing cover design input data, a few manufacturing constraints and an excerpt of a case report that outlines the results of the optimisation effort.

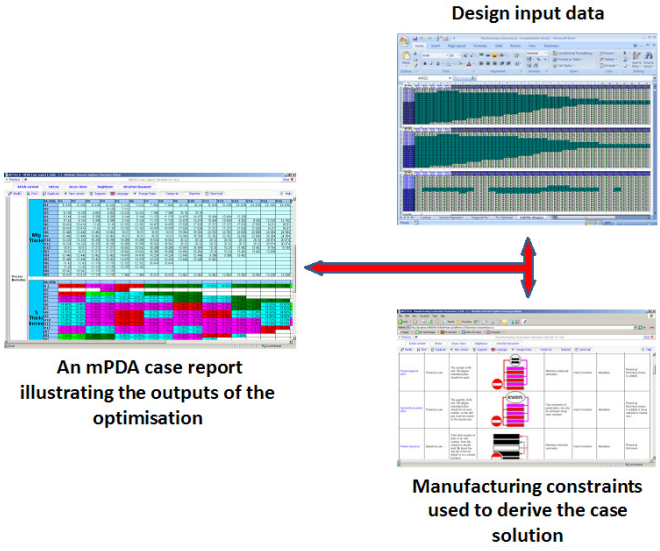

Figure 4.17: Case reports (left) and their relation with design inputs and underlying knowledge (Bermell-Garcia *et al.*, 2012)

The ply continuity optimization EKR has been annotated using the domain ontology and its constituent hierarchies, as introduced in Section 4.3.3.

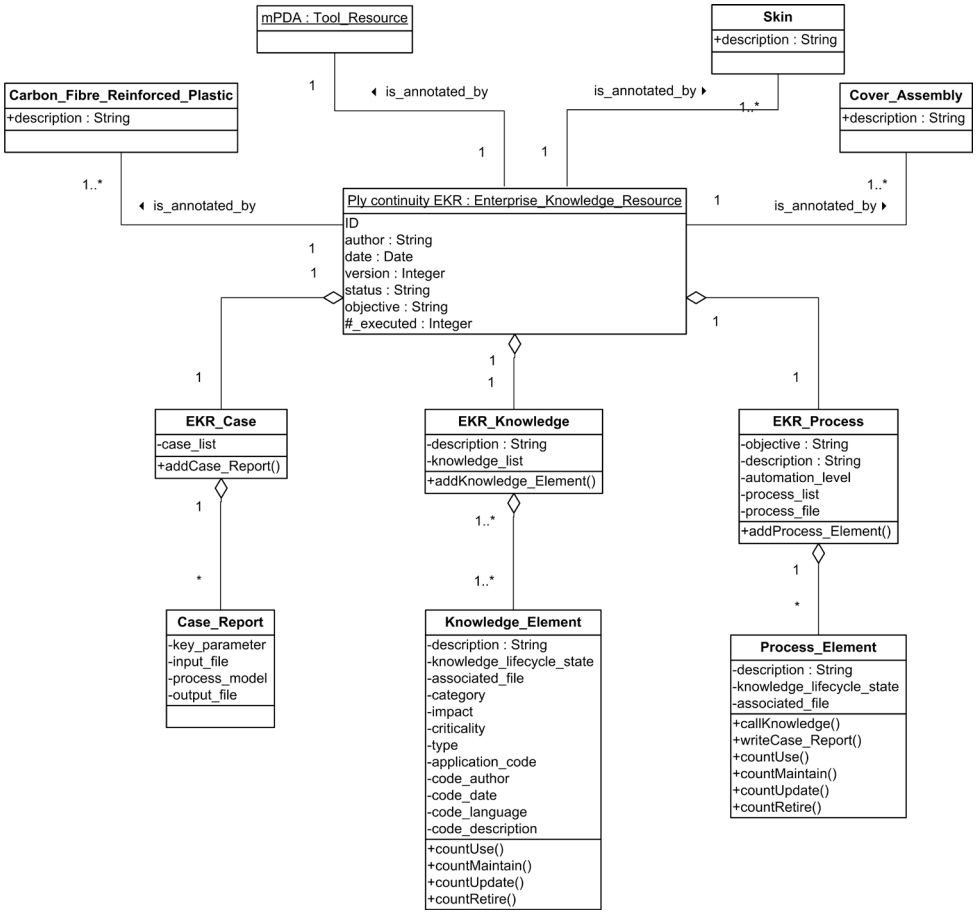

Figure 4.18: Annotation of the ply continuity optimization EKR

Figure 4.18 shows that a combination of product-, process- and resource-related tags can be used to annotate an EKR. The ply continuity optimization EKR has been tagged in AKM using the **Carbon_Fibre_Reinforced_Plastic** class (itself a subclass of **Material_Resource**), the **Skin** class (subclass of **Part**), **Cover_Assembly** (subclass of **Assembly**) and an *mPDA* object (instance of **Tool_Resource** class).

To summarize, the outcome of the development effort is a software architecture and an implemented solution, containing an EKR for ply continuity optimization. The central KLC ontology concept of Enterprise Knowledge Resource serves as an instrument to package an automated process with an associated knowledge-based tool (mPDA), knowledge (design and manufacturing constraints) and the history of cases generated using the knowledge and the tools.

4.3.5 Analysis & Delivery

The developed knowledge-based solution is able to optimize a composite wing cover conceptual design for ply continuity through blending of stacking sequences. The solution is automated. Furthermore, estimates of weight, manufacturability and cost are delivered as output of the solution. Pareto fronts can be plotted to visualize solution performance. As these results are confidential, they are not represented here.

A rough-order-of-magnitude estimate can be given regarding the non-recurring and recurring costs that have been necessary to a) develop and implement the ply continuity optimization EKR, including design and implementation of the software architecture, and b) will be necessary to maintain and/or expand the EKR approach. Roughly 6 man-months of development effort were spent on the non-recurring development effort. This has focused mostly on knowledge model construction, architecture design, building web services, server infrastructure and Model Center wrappers, and subsequent implementation of the EKR. A rough estimate of effort required to include existing application(s) as a new EKR within the eLBD/xLBD framework would be 1-4 man-weeks, depending on the state of maturity of knowledge, processes and application code. On the recurring cost side, the effort required to introduce new knowledge is fairly limited: up to 5 minutes for a knowledge base entry (including relations), up to 5 minutes to include application code if available beforehand, up to a few hours to develop application code if not available beforehand (though this is dependent on many factors and can increase significantly based on the language used, the experience of the developer, etc).

The recurring benefits of using a knowledge-based application are dependent on the selected application, but for the use case they are consistent with experience from other KBE research (Verhagen *et al.*, 2012): a reduction of design time from hours to a few minutes – roughly 95-99% – can be achieved. The benefits of using the EKR approach are hard to quantify, but can be qualified. First of all, using an EKR enables knowledge lifecycle management – knowledge can be kept up to date. Furthermore, the supporting framework enables availability of knowledge-based applications to a community of end users through the 'one-stop' EKR implementation, which in effect offers a service that is remotely accessible and remains traceable. The automatically generated case reports enable increased visibility and traceability of the analysis inputs, process and results. They enable subsequent analysis, for instance using case-based reasoning when multiple case reports are available. The knowledge necessary for optimization of ply continuity is gathered, classified and stored in the knowledge management tool (AKM) so that when a process is executed in an automated environment, the manufacturing decisions are available in the knowledge management tool. This moves the manufacturing input requirements outside of

the automated process execution loop, allowing the loop to run without any human interfaces. At the same time it enables the update of knowledge and tools as technology progresses.

4.4 Discussion of Results

A knowledge-based solution has been developed for the ply continuity optimization problem. It meets the case study requirements by being able to optimize ply stacking sequences for a full wing panel and delivering weight, manufacturability and cost estimates for conceptual wing panel designs. The solution uses knowledge and an automated process approach to deliver advantages traditionally associated with knowledge-based engineering applications. However, through the use of the EKR approach (as the cornerstone of the KLC ontology, and with its attendant models for the included classes), the developed KBS can cope with changing knowledge. As a result, knowledge can be effectively *utilized* and *maintained*.

With respect to the challenges related to knowledge usability and maintainability, the following issues have been addressed in the following manner:

- **Moving beyond black-box applications and ensuring transparency:** Knowledge is accessible and traceable through the use of the annotation metamodel, and through the use of case reports. Engineering tasks that use this knowledge and the resultant outputs are traceable and recorded systematically, so that the black-box phenomenon is avoided as much as possible. Through the use of the Enterprise Knowledge Resource approach, the process model and the associated inputs (the knowledge – design and manufacturing constraints – involved within the optimization process) and outputs (case reports) are made transparent. In particular, the knowledge elements can be accessed, used, maintained and updated throughout their life.
- **Task orientation:** knowledge involves a 'capability for effective action'. The capability for action is met by constructing an EKR that uses (sets of) constraint knowledge to execute a process. Effectiveness of the action is realized through task automation – through the use of Model Center, the EKR process model is automated, enabling faster optimization and evaluation of alternatives for composite wing covers.
- **Expert / end user involvement:** Through the EKR approach – and in particular the **Knowledge** class – end users can identify, use, interact with and if necessary, maintain or update the relevant knowledge that is used to design manufacturable composite wing covers.

5 Manufacturing Case Study: Composite Wing Cost Modelling & Estimation

This chapter describes the development of a knowledge-based application that supports manufacturing cost evaluation of composite wing covers.

This case study is the second of a set of three case studies. This case study focuses on the manufacturing domain of the aircraft lifecycle. Together, the three case studies will shed light on how the overall research objective can be achieved, with emphasis on the latter part of the objective: "Support consistent formalization, use and maintenance of changing knowledge within aircraft lifecycle phases *to improve domain-specific modelling, execution and control of engineering tasks*". The case studies also offer a practical perspective on the following research questions:

- How can knowledge change be accommodated during knowledge-based application development?
    - Which models are required and how do these models help to accommodate knowledge change?

The following section introduces the problem for this specific case study in the aircraft manufacturing domain. After this, the theory contributions are applied to the case study: the Knowledge Lifecycle model is used to identify knowledge change, the engineering task is analysed and the KNOMAD methodology steps are planned out. A solution has been developed for the case study problem; development and implementation are discussed in detail in Section 5.3. Validation of performance with respect to the case study objective(s) and requirements is briefly indicated in Section 5.3.5: Analysis & Delivery. The case study concludes with a discussion of the results within the context of the dissertation objectives and contributions to theory.

This case study presents an approach to support manufacturing cost modelling and estimation for composite wing components. A solution has been developed on the basis of the KLC ontology, using the KNOMAD methodology. As in the previous case study, the solution supports the deployment and use of knowledge as an element in modular knowledge packages (the previously introduced Enterprise Knowledge Resources) that are managed in a central knowledge repository. These EKRs can be deployed to support the manufacturing cost modelling and estimation task. The developed solution supports manufacturing cost evaluation of product concepts at early stages of the design process, while offering the opportunity for through-life knowledge support - a vital requirement given that the manufacturing knowledge underlying the cost model(s) is subject to change.

5.1 Case Study Context and Challenges

The case study concerns a legacy cost model that has been developed at a large aerospace OEM to address current issues on cost estimation of conceptual designs of composite wing cover parts and assemblies.

In literature, research on cost modelling is rich and varied. A number of authors discuss and categorize current approaches to cost estimation. Niazi *et al.* (2006) give an overview of cost modelling and estimation techniques. Though the categorization may be disputed, the overview is fairly comprehensive.

Table 5.1: Product cost modelling and estimation techniques (adapted from Niazi *et al.* (2006))

<table>
<tr><td colspan="4">Product cost estimation techniques</td></tr>
<tr><td rowspan="6">Qualitative cost estimation techniques</td><td>Intuitive cost estimation techniques</td><td colspan="2">Case-based systems</td></tr>
<tr><td rowspan="3"></td><td rowspan="3">Decision support systems</td><td>Rule-based systems</td></tr>
<tr><td></td><td>Fuzzy logic systems</td></tr>
<tr><td></td><td>Expert systems</td></tr>
<tr><td>Analogical cost estimation techniques</td><td colspan="2">Regression analysis model</td></tr>
<tr><td></td><td colspan="2">Back propagation neural network model</td></tr>
<tr><td rowspan="6">Quantitative cost estimation techniques</td><td>Parametric cost estimation techniques</td><td></td><td></td></tr>
<tr><td rowspan="5">Analytical cost estimation techniques</td><td colspan="2">Operation-based cost models</td></tr>
<tr><td colspan="2">Break-down cost models</td></tr>
<tr><td colspan="2">Cost tolerance models</td></tr>
<tr><td colspan="2">Feature-based cost models</td></tr>
<tr><td colspan="2">Activity-based cost models</td></tr>
</table>

Another perspective on cost modelling (Curran *et al.*, 2004; Feldman and Shtub, 2006; Price *et al.*, 2006; Newnes *et al.*, 2008) highlights three approaches, namely analogous, parametric and bottom-up cost modelling. Curran *et al.* (2004) present a matrix of comparative assessment for these methods, which is given here in Table 5.2. Notably, one of the subsets of bottom-up modelling is physical process modelling, which focuses on the time required to carry out work (Curran *et al.*, 2004). This principle is used in the cost modelling approach outlined in this case study. The physical process modelling technique as discussed by Curran *et al.* (2004) bears great similarity to the operation-based cost modelling technique identified by Niazi *et al.* (2006).

Table 5.2: Assessment matrix for traditional cost estimation methods (Curran *et al.*, 2004)

Approach	Advantages	Disadvantages
Bottom-up costing	Cause and effect understood Very detailed estimate	Difficult to develop and implement Substantial, detailed expert data are required Requires expert knowledge
Analogous costing	Cause and effect understood More easily applied than bottom-up method	Appropriate baseline must exist Substantial, detailed data are required Requires expert knowledge
Parametric costing	Easiest to implement Non-technical experts can apply method Uncertainty of the forecast is generated Allows scope for quantifying risk	Can be difficult to develop Factors might be associative but not causative Extrapolation of existing data to forecast future products including new developments might be unwarranted

Curran *et al.* (2004) also distinguish between 'traditional' and 'advanced' estimating approaches. The three approaches mentioned in Table 5.2 are deemed to be traditional, whereas advanced estimating approaches include the use of feature-based modelling, fuzzy logic, neural networks, uncertainty modelling, and data mining. Curran *et al.* (2004) also introduce the genetic causal cost modelling approach to address the need for a more scientifically based methodology for cost estimation.

Newnes *et al.* (2008) add to this perspective by considering cost estimation approaches as being 'generative' or 'parametric'. In the generative process, the cost estimation builds upon the data that is gathered during the design process. Consequently, the accuracy of the costing estimate depends on the level of data detail. In parametric approaches, estimates are “achieved based on past experience, using findings from past products and estimating the expected cost” (Newnes *et al.*, 2008). Parametric approaches distinguish themselves by the use of cost-estimating relationships (CER).

Newnes *et al.* (2008)’s categorization relates closely to the level of fidelity of the cost estimation approaches; as Price *et al.* (2006) indicate, “analysis fidelity relates to the degree of detail and accuracy contained in a given analysis model”. Price *et al.* (2006) distinguish three levels of fidelity. Low fidelity models use simple equations and look-up tables, and frequently do not have associations with geometric models. Medium fidelity models use some form of linear analysis in combination with geometric model information and high fidelity models contain a lot of detail while modelling non-linear behaviour. When looking at the issue of

fidelity from a more multidisciplinary perspective, a number of research gaps can be identified (adapted from Price *et al.* (2006)) – see Table 5.3.

Table 5.3: Disciplines versus fidelity (adapted from Price *et al.* (2006))

		Analysis Discipline						
		Aerodynamics	**Structures & weights**	**Noise**	**Geometry**	**Cost**	**Manufacturing**	**Maintenance**
Fidelity	**Low**	Empirical methods (e.g. EDET)	Empirical methods (e.g. FLOPS)	Empirical methods (e.g. FLOPS)	Vehicle Sketch Pad (VSP)	Piano, SEER, Price	**GAP**	**GAP**
	Medium	Vortex Lattice (e.g. WINGDES)	Basic structures (e.g. ELAPS)	Basic Noise (e.g. ANOPP)	**GAP**	**GAP**	**GAP**	**GAP**
	High	CFD (e.g. FUN2D, Fluent)	FEM (e.g. NASTRAN)	Advanced Noise (e.g. AVATAR)	High-end CAD systems (e.g. CATIA)	**GAP**	Virtual Factory Simulations (e.g. DELMIA)	**GAP**

The modelling approach used in this case study moves from low to medium fidelity: it combines relatively detailed geometric model information with linear analysis and look-up tables. The resulting cost model can be classified as bottom-up, as it uses a physical process modelling approach to estimate costs. In its legacy form, the cost model is spreadsheet-based. For a given product with adjustable characteristics (e.g. material and geometry), it allows for the estimation of time and cost associated with manufacturing process options. The model has been developed for a range of composite materials and manufacturing processes.

The core cost modelling approach is to use geometry input, manufacturing parameters and rules representing manufacturing processes and underlying sub-processes to arrive at estimates for process times and costs, which are subsequently added to arrive at totals for time and cost. The approach can be schematically summarized into a calculation process with a number of standard elements, as illustrated in Figure 5.1.

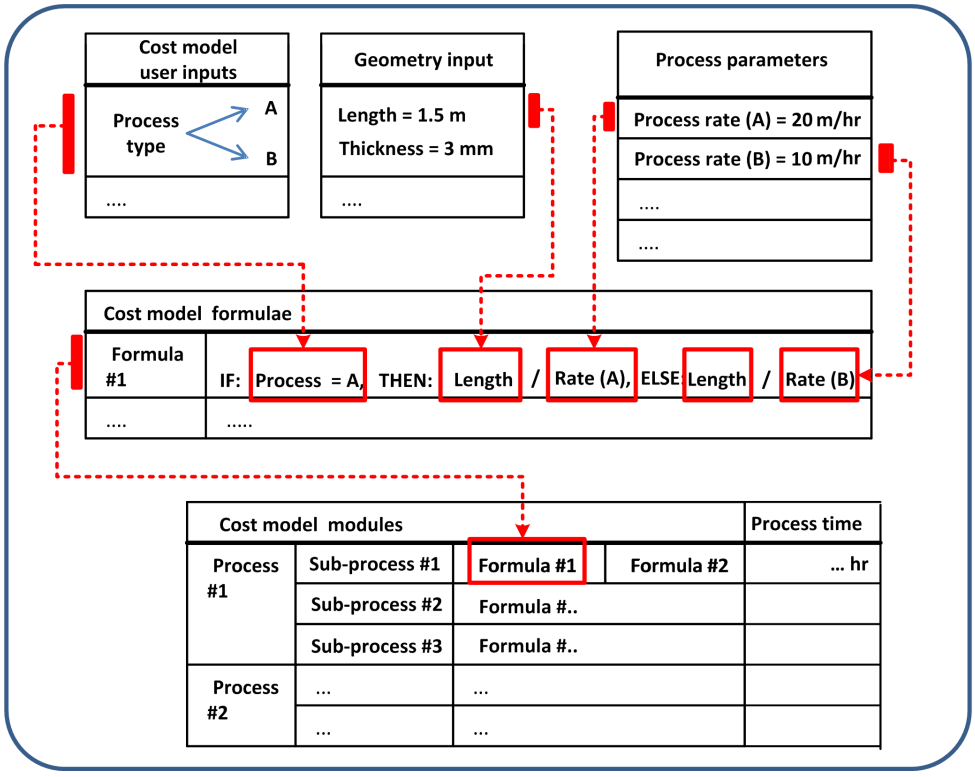

Figure 5.1: Standard cost modelling approach

When analysing the full cost model, the standard model elements can be generalised and more properly classified. The following basic cost model elements are identified:

- **Cost model parameters:**
  - **Process parameters:** These are single values associated to process parameters expressing processes with constant time requirements for a given product and process combination (e.g. autoclave time) or production rates for a given product and process combination (e.g. cut rate). In most cases, they come from existing manufacturing cost estimation data and tools. For the remaining cases, cost estimators have identified the best possible parameter values based on expert assumptions and estimation.
  - **Geometric parameters:** these parameters express the product geometry. This geometry can be added to the model in two ways. The first and preferred option is to import the geometry into the spreadsheet from CATIA using a conversion

from XML data into so-called 'fishtail' plots of product geometry in Excel. In this way, up to date product geometry is imported into the cost model. The alternative is to manually enter product geometry values, but entering this data requires more effort, the process is prone to error and data can become out of date.

- **User inputs:** Users can choose some inputs to the cost model to reflect choices in manufacturing processes (e.g. automated tape laying or hot roll forming, or the applied amount of non-destructive testing) and product design (e.g. the number and position of spar caps). User inputs are usually implemented as value choices that modify the behaviour of cost model formulas through the use of "IF-THEN" rules.
- **Model equations:** These are formulas that use process parameters, geometric parameters and/or user inputs to compute time and cost estimates for each manufacturing process.

The cost model is discussed in more detail in section 5.3.1: Knowledge capture, which includes examples for geometric input, a full sub-process and estimation output.

The legacy cost model has a number of drawbacks. First of all, the development of the cost model has made it very difficult to manage its complexity, as the evolved model consists of many interrelated formulas and inputs that are insufficiently classified. An associated drawback is that the inputs used for the model are not maintained on a shared base, but are instantiated for each version of the model: each user has a 'unique' spreadsheet model. If a user makes changes to parameter values to adjust for new knowledge, these changes are not shared with other cost model instances in the business. Maintainability of cost model knowledge is a significant problem.

Another drawback of the legacy cost model is its rather inflexible, monolithic nature: the current spreadsheet implementation is focused on certain process-product combinations and does not allow for easy mixing of different materials or sub-processes. A possible route to address this would be to enable the assembly of cost model elements that are stored in a managed environment. Also, the cost model is maintained principally by only two persons, as the complexity of the model precludes more direct governability by the end users (even though the latter can still configure user inputs, variation in the process parameters and equations is discouraged). This is a business risk, as the full set of required modelling knowledge resides with only two persons in the organization. Both the inflexible nature of the cost model and the lack of more direct control by the end

user provide significant challenges for user involvement and transparency of knowledge. Also, in its evolved form, the cost model can be typified as a 'black-box' application. Because of its complexity and its distribution over multiple spreadsheets and spreadsheet tabs, a user effectively has no other option than to trust the model output. The user must go to considerable lengths to retrieve the original knowledge sources behind the implemented formulas and parameters; the supporting informal knowledge is also very hard to find.

The assumptions, inputs, operations and outputs of the cost model are not managed from a life-cycle perspective. There are currently little to no provisions for explanation of the rationale behind assumptions. Furthermore, the cost model knowledge, embodied in model inputs, rules and outputs, changes during the lifetime of the cost model (see Section 5.2.1), but these changes are not stored, let alone tracked. This lack of knowledge maintainability is a significant stumbling block in the learning process towards composite component production and its associated cost estimation.

The consolidated objective of the case study is to develop and implement a proof-of-concept knowledge-based solution for cost modelling and estimation of composite wing cover manufacturing processes. The following requirements must be met:

1) The solution must be able to support end users in composition, use and control of a cost model;
2) The solution must give the possibility to quickly estimate cost for composite wing covers;
3) The solution must be automated to the fullest extent possible.

Validation with respect to these requirements is performed in Section 5.3.5: Analysis & Delivery.

The introduced problem is related to knowledge change and consequently to research challenges regarding knowledge usability and maintainability. The following issues will be taken into account in the case study:

- **Moving beyond black-box applications and ensuring transparency:** As mentioned before, the cost model can be typified as a 'black-box' model: the knowledge is difficult to access, inspect and maintain. To support knowledge maintenance, it is necessary to move beyond the current black-box implementation by supporting categorization, accessibility, traceability and subsequent sourcing of knowledge. The solution must enable the storing, justifying and updating of cost model knowledge elements and must support recording of previous versions of the cost

model. To ensure transparency, it should be clear which knowledge is involved within the cost modelling and estimation solution, which inputs are necessary, which steps are taken within a process and which outputs are generated. The solution must enable a standard approach of costing parts.

- **Task orientation:** knowledge implies a 'capability for effective action'. The capability for action can be met by explicitly associating sets of knowledge with functional tasks, i.e. the estimation of costs related to specific products and/or manufacturing processes. The solution will be designed to improve upon the legacy black-box implementation of the cost modelling and estimation capability. To achieve this, the solution will not replace the use of spreadsheets to compute the cost of components. However, it manages the knowledge driving the cost model and deploys it to a working spreadsheet from which users can understand the rationale of the computed cost and access the underlying knowledge.
- **Expert / end user involvement:** End users must be able to identify, use, interact with and if necessary, maintain or update the cost model. The primary aim of the resulting capability is to be able to estimate costs by 'running' cost models using trustworthy and up-to-date knowledge.

Through these considerations, the case study contributes to validation of the overall research contributions to theory. This is discussed in Section 5.4: Discussion of Results.

5.2 Application of Theory to Manufacturing Case Study

Before developing a solution, this section acts as an intermediate step by applying the developed theory to the case. First, the Knowledge Lifecycle Model is applied to identify knowledge change for the cost modelling and estimation task. This task is subsequently analysed in support of further application of the KLC ontology to solution development in the Results section. Finally, it will be shown how the KNOMAD steps will be applied to this case to guide the subsequent solution development in Section 4.3: Results.

5.2.1 Application of Knowledge Lifecycle Model: Identifying Knowledge Change

The research problem addressed in this use case emerges from the difficulties of coping with the complexity added to the cost model during its evolution. This increase in complexity is illustrated in Figure 5.2.

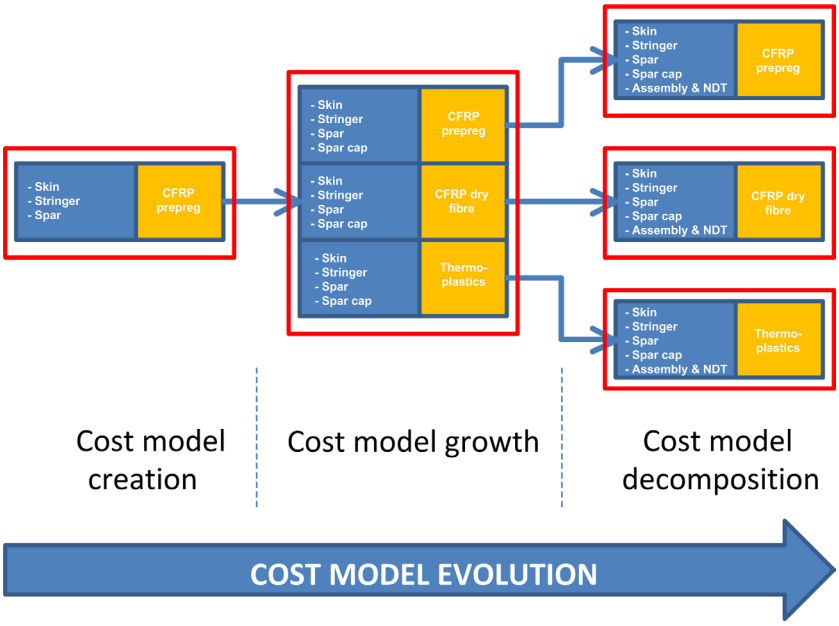

Figure 5.2: Cost model evolution

From this Figure, the following cost model stages can be observed:

- **Cost model creation:** An initial cost model using manufacturing process aspects that influence the cost of composite wing covers was captured within a spreadsheet. The information used was not completely new since some of the parameters and their values came from existing cost estimation data and tools.
- **Cost model growth:** The cost model gained enough relevance and trust among cost engineers, resulting in an expansion of scope. Further developments on the cost model allowed users to consider costs for various product-process combinations using up to 5 different composite materials / material types (only three are shown in Figure 5.2). The necessary knowledge was distributed across spreadsheet tabs. In this growing process, new knowledge was added to the model. However, existing pieces of knowledge were reused across the material tabs. At that stage, over 400 process-related parameters were at the basis of the model, augmented with some 30 parameters for which the values were chosen by the user (see Section 5.3.1 for more detail). These parameters were driving hundreds if not thousands intermediate calculations, frequently with unique formulas to take into account user configurations, to arrive at process time and cost estimates. The model had become a very complex web of knowledge interactions, in which most knowledge

elements had become tacit in nature as these elements would require the explanation of an expert to make sense to outsiders.

- **Cost model decomposition:** Further enrichment of the cost model (addition of materials, products and processes) forced its developers to split it into pieces. Other reasons to decompose the model could be found in the need to distribute it to different users responsible for the cost estimation for different composite materials and material types. In this process, some of the knowledge was classified and distributed to its consumers. However, a significant risk of inconsistency and duplication of data, information and knowledge emerged, as well as difficulties with consistency in data fidelity. Furthermore, the complexity of the spreadsheet made it difficult for management and advanced use by anyone other than the creators.

As can be seen from these stages, knowledge change for this case study is embodied in cost model change with respect to data, information and knowledge. Data changes when parameter values, for instance for manufacturing process (steps) such as bagging, curing, cutting or non-destructive testing, are changed to reflect updated process specifications (i.e. *update*). The data context also changes (i.e. *maintain*), as more material types and manufacturing processes are added when the cost model is expanded. This constitutes an information change in the cost model. Knowledge contained in the model changes in a number of ways. First of all, the equations and rules expressing the core knowledge about a manufacturing process may be *updated* as processes change over time. Furthermore, the context of the knowledge changes (*maintain*). Finally, the capability for effective action- cost estimation, in this case - at first grows, but later reduces due to model complexity.

Similar to the design case study, the preceding discussion qualitatively shows that knowledge is subject to change with respect to the cost modelling and estimation task. Quantification of this change using the Knowledge Lifecycle Model concepts of knowledge actions has not been performed due to the historical nature of the cost model.

5.2.2 Application of KLC Ontology: Task Analysis

The cost modelling and estimation task is given in Figure 5.3 as an A-0 IDEF0 diagram. For the legacy process, the cost model is implemented in Excel. A cost engineer can use the cost model in conjunction with geometry and process parameters to produce cost estimates for specific combinations of manufacturing processes and products. To do so, design choices have to be indicated in the model by the user.

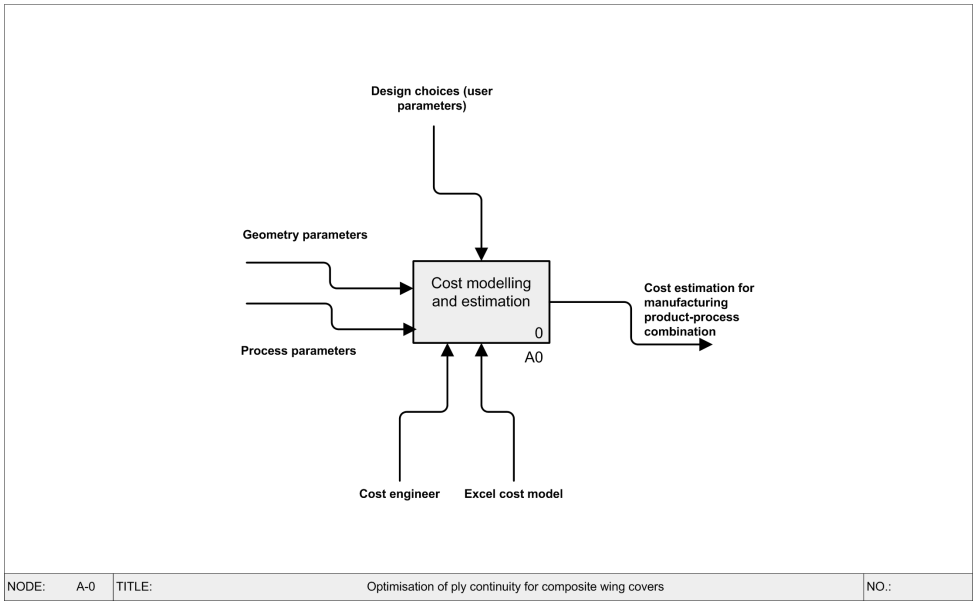

Figure 5.3: IDEF0 A-0 diagram for cost modelling and estimation task

The task is split up into several subtasks, which are represented in Figure 5.4 as an A0 IDEF0 diagram. The first subtask is preparation: the cost engineer retrieves a generic cost model from the company repository – a project hard-drive for the legacy process. The engineer then manipulates the model by specifying or importing product geometry and making design choices (e.g. which manufacturing process for production of a specific part). The output of this subtask is a specified cost model for a particular product-process combination. The final subtask is the generation of a cost estimation report, which typically consists of the construction of tables and/or graphs based on the cost model output for use in company reports.

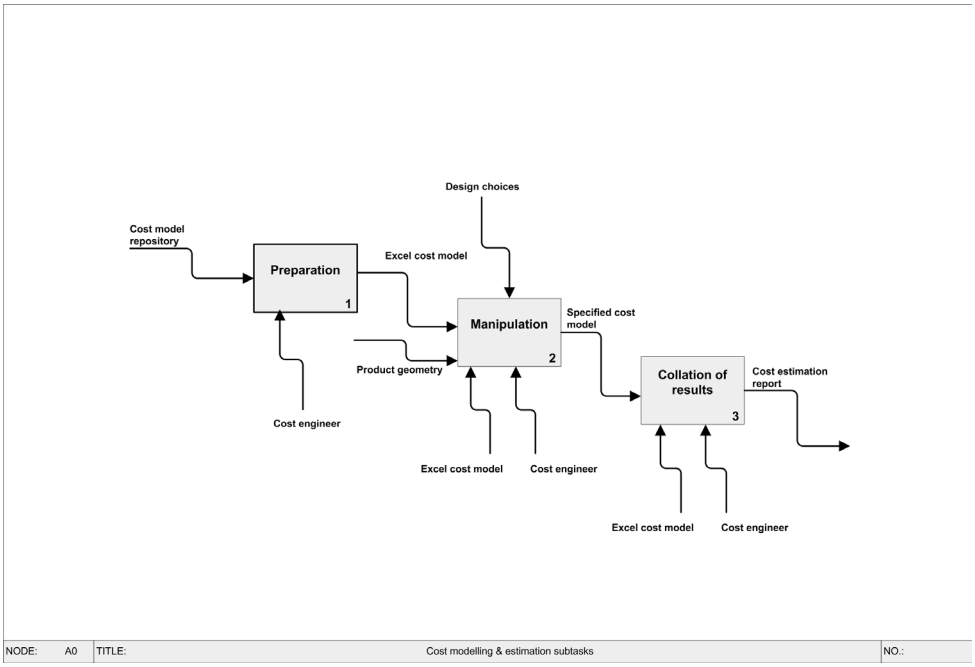

Figure 5.4: IDEF0 A0 diagram for cost modelling and estimation subtasks

The actual estimation of cost relies on cost functions for manufacturing process-product combinations that are implemented in the cost model. This is explained further in Section 5.3.1.

5.2.3 Application of KNOMAD: Solution Approach

The KNOMAD methodology as discussed in Section 3.3 is adopted to develop a knowledge-based application for the case study problem. The KNOMAD steps relative to this case study are shown in Figure 5.5. This figure shows the main KNOMAD steps (Knowledge Capture & Identification of Knowledge Change; Normalisation; Organisation; Modelling & Implementation; Analysis & Delivery) with the associated activities that are required for this particular case study.

In the first step (Knowledge Capture & Identification of Knowledge Change), the justification for and scope of the knowledge-based application is established, followed by capture of the knowledge and process elements. The cost model parameters and equations as well as the inputs to the problem are of particular interest. As knowledge change for this case study has already been considered in Section 5.2.1, this activity is not repeated. For the second step (Normalisation), the focus is on checking data quality and establishing input and output formats. The third step (Organisation) considers development of a domain ontology that holds the relevant concepts and relationships for this particular case study. It is split up into three parts: generation of product, process and resource class

diagrams. The fourth step (Modelling & Implementation) concerns the development of models (in the Modelling step), architecture and solution (in the Implementation step). As part of the ontology-based approach, the task and domain ontologies are implemented in AKM to support the developed solution. Finally, the Analysis and Delivery steps are combined into one: performance of the solution is assessed relative to the requirements, and the costs and benefits of the solution are explored.

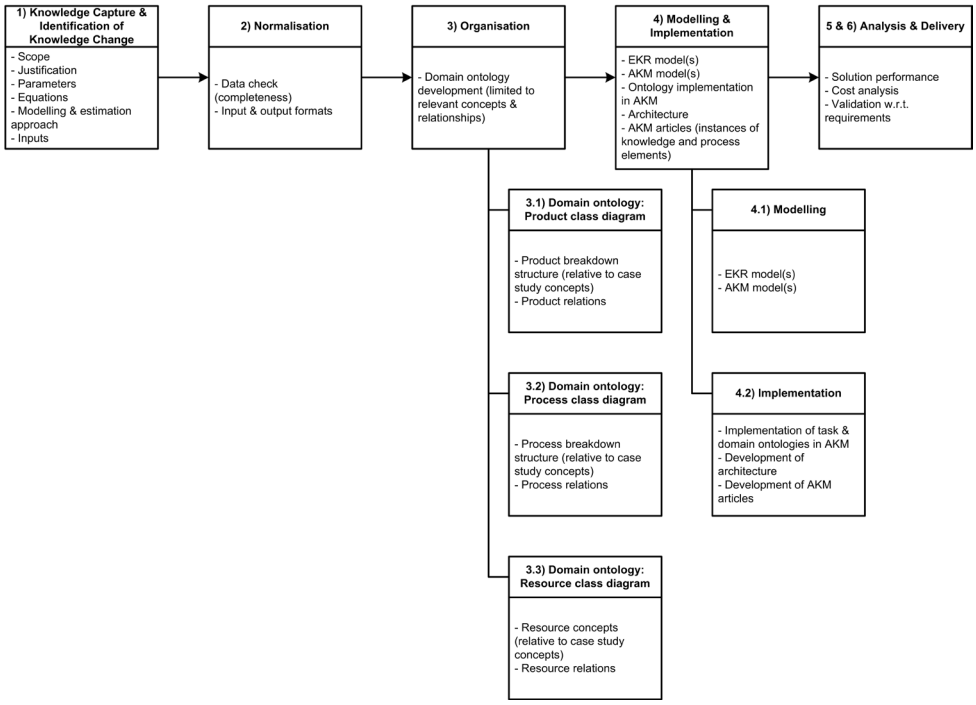

Figure 5.5: Application of KNOMAD to manufacturing case study – flow chart

5.3 Results

The next section describes the development of a knowledge-based application for the cost modelling and estimation problem. The proof-of-concept solution can cope with data, information and knowledge change and addresses associated issues related to knowledge usability and maintainability. The development of the solution is based upon application of the revised KNOMAD methodology. Furthermore, the principles and concepts of the Knowledge Lifecycle ontology are used. The following sections are compliant with the KNOMAD steps.

5.3.1 Knowledge Identification & Capture

In this Section, an analysis will be performed on part of the full cost model focusing on carbon fibre-reinforced plastic (CFRP) wing top cover manufacturing. As introduced before, the core cost modelling approach uses geometry input, manufacturing parameters and rules representing manufacturing processes and underlying sub-processes to arrive at estimates for process times and costs.

Table 5.4: Example cost modelling approach: T-stringer Production

	Sub-Process	Detail Process	Calculation				Baseline	Target
T Stringers production	Strgr ATL (2D)	Setup ATL & Load Tool	Set Time		x	Constant		
		Lay Slab	Weight		/	Deposition Rate		
		Unload ATL	Set Time		x	Constant		
	Strgr ATL (2D) Cut & Kit	Cut Slab into Planforms	Slab Perimeter		/	Cut Rate		
		Kit Planforms	N Stringer pieces		x	Constant		
	Strgr Hot Roll Form	Setup & Soak Hot Roll Form	N Stringers		x	Constant		
		Feed Stringers	Stringer Weight		/	Roll Rate		
		Unload Hot Form	N Stringers		x	Constant		
	Strgr Hot Form	Load Planforms to Tool	Stringer Length		/	Load Rate		
		Hot Form Cycle	Form Cycles		x	Hot Form Cycle		
		Unload Hot Form	N Stringers		x	Constant		
	Stringer Profile Shape	Position Blade	N Stringers		x	Constant		
		Make Final Profile	N Stringers		x	Constant		
		Insert Noodle	Length		/	Rate		

A straightforward example of the general cost modelling approach is shown in Table 5.4. This table shows the basic steps involved at estimating the costs for the production of CFRP T-stringers, which are parts (potentially) involved in composite wing cover manufacturing. This example shows the main steps in arriving at process time estimates. At the left-hand side, the overall process (T-stringer production) is subdivided into sub-processes, which are subdivided themselves into two to three detailed processes. To initiate the analysis, a set of geometry parameters (e.g. weight, slab perimeter, number of stringer (N stringers), length) are derived from geometric models that are coupled with the cost model spreadsheets. An example of imported geometry data for a composite wing top cover is given in Figure 5.6, which shows wing cover geometry data on the right hand side and intermediate calculations for thickness, weight and processing time on the left hand side[§]. Furthermore, some typical process parameters are taken from a set of known or estimated parameters (e.g. set time for Automated Tape Laying (ATL) tool load and unload): example parameters are given in Figure 5.7, though values have been excluded for confidentiality. The initial calculation parameters derived from geometry or from known parameter sets are allocated per detailed process (as given in the left-most column under the

[§] The data values are illegible on purpose, to maintain confidentiality.

'Calculation' heading)**. They are then multiplied with or divided by applicable manufacturing process constants or rates, both in baseline and target forms. The baseline form represents process performance that is currently achieved, whereas the target form represents the anticipated performance by the time of production start. By and large, the rates and constants in the right-most 'Calculation' column (e.g. deposition rate, cut rate) are taken from parameter sets that almost invariably lack justification and traceability of data.

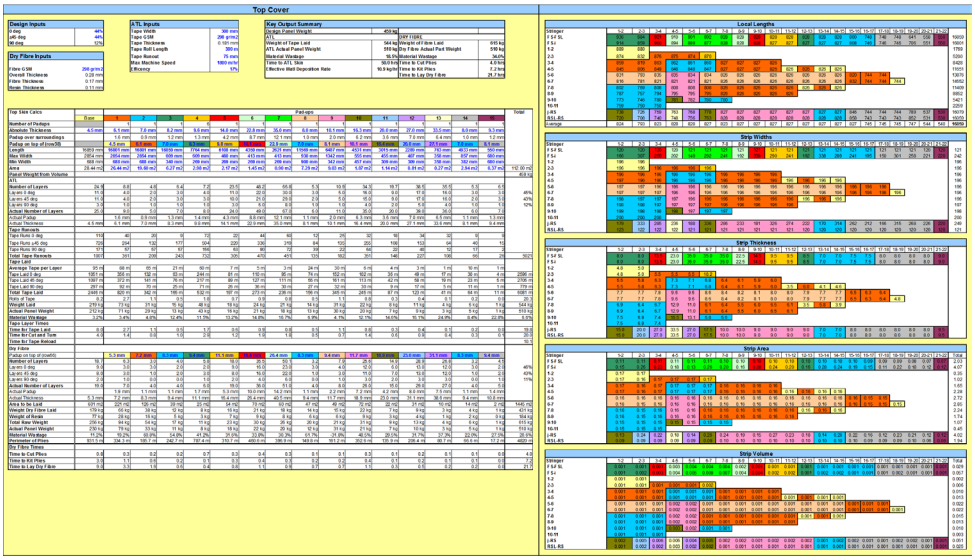

Figure 5.6: Example of imported geometry data for a composite wing top cover

** The actual values for the calculation parameters in Table 5.4 have been excluded for confidentiality reasons.

Top Skin Inputs				Value				Source
Skin Average Thickness	Top_Skin_Prepreg_Avg_T					m		?
Bottom Skin Inputs								
Skin Average Thickness	Btm_Skin_Prepreg_Avg_T					m		?
Knowledge								
Material			Min	Baseline	Max	Probability	Target	Source
Thickness of the Prepreg Material		Prepreg_Matl_T						EMEV
		DryFibre_Matl_T						EMEV
	m	Core_T						
	m	Prepreg_Tape_Runout						EMEV
	kg/m3	Fibre_Density						approx Sta
	kg/m3	Resin_Density						approx Sta
	kg/m3	Prepreg_Density						approx Sta
	kg/m3	Core_Density						
		N_Ribs						
	m	Hot_Form_Bed_L						
		N_Sections_Form_Cycle						A350
	m	Manhole_Circum						EMEV
	m	T_Stringer_Avg_T						EMEV
		N_Fast_Strgr						Fasteners
		N_Fast_Rib_Foot						Fasteners
		N_Fast_Rib_Post						
	m	Rib_Post_Avg_W						
	m	Rib_Post_Avg_T						
		Rib_Feet_Avg_W						

Figure 5.7: Example of process parameters

The results of this approach are time estimates for the detailed process steps. This is followed by a number of calculations to arrive at cost estimates (e.g. through the simple step of multiplication of process times with labour rates), and supplemented with logistical cost estimation (e.g. capital outlay for a production run of certain size).

The format for cost estimation output is shown in Figure 5.8. Confidential information (i.e. geometry and cost estimation values) has been excluded.

Key Geometry			
Plan	A	Status	11
Top Skin		Top Stringers	
Skin Length		Stringers Length	
Skin Root Width		N Stringers	
Skin Tip Width		Stringer Type	
Skin Area		Stringer Weight	
Skin Weight			
Bottom Skin		Bottom Stringers	
Skin Length		Stringers Length	
Skin Root Width		N Stringers	
Skin Tip Width		Stringer Type	
Skin Area		Stringer Weight	
Number of Manholes			
Cover Weight			
Front Spar		Rear Spar	
Spar Length		Spar Length	

Manufacturing Inputs			
Stringer Manufacture	Standard	Cover Bagging Style	Bag
Spar Cap Manufacture	Standard	Skin Cure pre Strg Int	Yes
Top Cover Spar Cap	Front	Number of Covers per cure	1
Btm Cover Spar Cap	Both	Extra NDT of Skin	Yes
Stringer Batch Size	8		
Manual or Fishtail Data	Fishtail		
Hourly Rate Options			
Hourly Rate Used	Calculated	Monthly Production Rate	80
Facility Dedicatation	Shared	Aircraft per annum	0
Material Inputs			
Generic			
Dry Fibre Fibre Content	60.0%	Movement Allowance	10%
Dry Fibre Knockdown	15.0%	Min_Load_Factor	20%
Dry Fibre Allowance	50 mm		

Summary per wing	Labour Hours	Labour Cost	Material Costs	Total Cost	Cycle Time	CAPEX	Material Waste	Weight
Top Skin Prepreg								
Bottom Skin Prepreg								

Figure 5.8: Cost estimation output format

5.3.2 Normalization

To ensure knowledge quality and compliance to a standard, measures have been performed on the captured knowledge:

- **Traceability & Ownership:** the captured knowledge has been recorded using a pre-defined format that has been designed and agreed upon during the research process. The format is implemented in a knowledge management tool (the previously introduced Ardans Knowledge Maker, or AKM – see Section 4.3.4.2) and has two main components. First, an informal representation of a cost model element is recorded; it consists of a natural language description explaining its applicability and properties and is accompanied by any relevant illustrations. It also includes metadata, i.e. data about data, for instance ownership, authorship, date of creation and date of last interaction. Secondly, a formal representation of a model element either records the data for a parameter or the Excel code for a model equation. An example of a cost model element is illustrated in Figure 5.9.

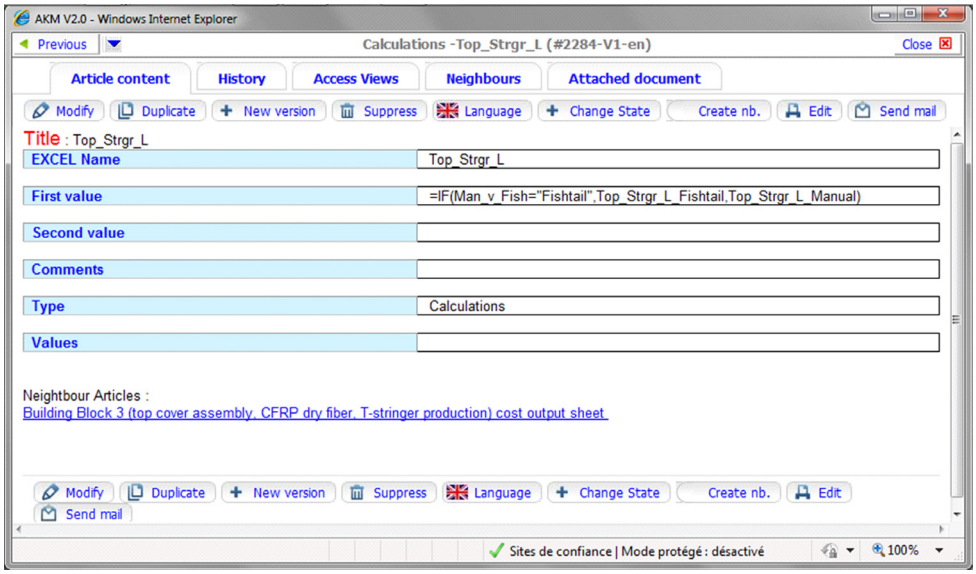

Figure 5.9: Example of cost model element stored in AKM

For this specific element the formal representation is used to contain Excel code for calculation of a certain stringer length. This code uses a user choice between manual geometry and fishtail (imported) geometry for looking up a geometry parameter value.

- **Accuracy & reliability:** the pre-existing Excel cost model has been used as a baseline for the solution development. The required manufacturing parameters (input, process and output) have been identified and captured. The resulting standardized formats are further explained in Section 5.3.4: Modelling and Implementation.

5.3.3 Organisation

The next step is to provide a knowledge structure that can be used to store and represent the captured knowledge.

To achieve this, it is necessary to construct a domain-specific set of concepts and relationships: a domain ontology. To elicit the applicable concepts and relationships for the aerospace composite manufacturing ontology, various sources have been employed. First of all, a small number (N = 4) of experts from the manufacturer have been interviewed. The results have been augmented by analysis of company sources, including the original cost model. This model has been organised to conform to a very basic classification after its evolution into several decomposed spreadsheets. This decomposition was made on the basis of the composite manufacturing technology, which incorporates material type and material processing form: similar models exist for different materials and forms, e.g. carbon-fibre reinforced plastics (CFRP) prepreg, CFRP dry fibre or CFRP sandwich. Besides this elementary subdivision, the cost models themselves are organised according to a manufacturing breakdown structure and manufacturing processes. The breakdown structure is a hierarchical breakdown of assemblies and products. In the cost model, similar calculations are performed for different products, for instance the bottom and top covers of the wing box. Furthermore, similar manufacturing processes are employed for parts of these products. For instance, the T-stringers from Table 5.4 are used in both bottom and top cover production.

In this section, excerpts of the domain-specific class hierarchies are given to explain how the manufacturing domain ontology is composed. The domain ontology development is not exhaustive and can be extended using considerably more detailed concept representations, but it is considered sufficiently complete for the purposes of this case study, i.e., for use in annotation of the solution (EKRs). Note that the manufacturing domain ontology is based on the PPR paradigm as implemented in the KLC ontology (see Section 3.2.3.3). It consequently shares the top-level classes and relationships given in Figure 3.7 and Table 3.7, which are also the basis of the top-level structures for the design and maintenance domain ontologies.

The PPR paradigm as contained in the high-level concepts of the KLC ontology (**Product**, **Process** and **Resource**) is extended for this case study in the form of the manufacturing breakdown structure, manufacturing processes and composite manufacturing material resources. These concepts can be used to organise and annotate the cost model elements. As in the design domain, these concepts of the KLC ontology have been extended into domain-specific class hierarchies – in fact, most of the subclasses and relations from the design domain ontology have been maintained. Figure 4.10, Figure 4.11 and Figure 4.12 can therefore be seen as a baseline for the manufacturing domain ontology. However, the manufacturing

domain ontology is extended with a few classes and attributes when compared to the design domain ontology.

The first domain-specific extension concerns the manufacturing breakdown structure as captured in the **Product** class hierarchy (Figure 5.10). It retains a large number of classes previously introduced as part of the design domain ontology (Section 4.3.3), including the **Assembly** class and its subclasses (with the **Wing Box Assembly** being a particularly important example) and the **Part** class (including **Spar, Rib, Skin** and **Stringer**). Only a few additions were made to these classes to incorporate case study-specific concepts: classes for the front and rear spar assemblies, and the **Boom** and **Supports** parts.

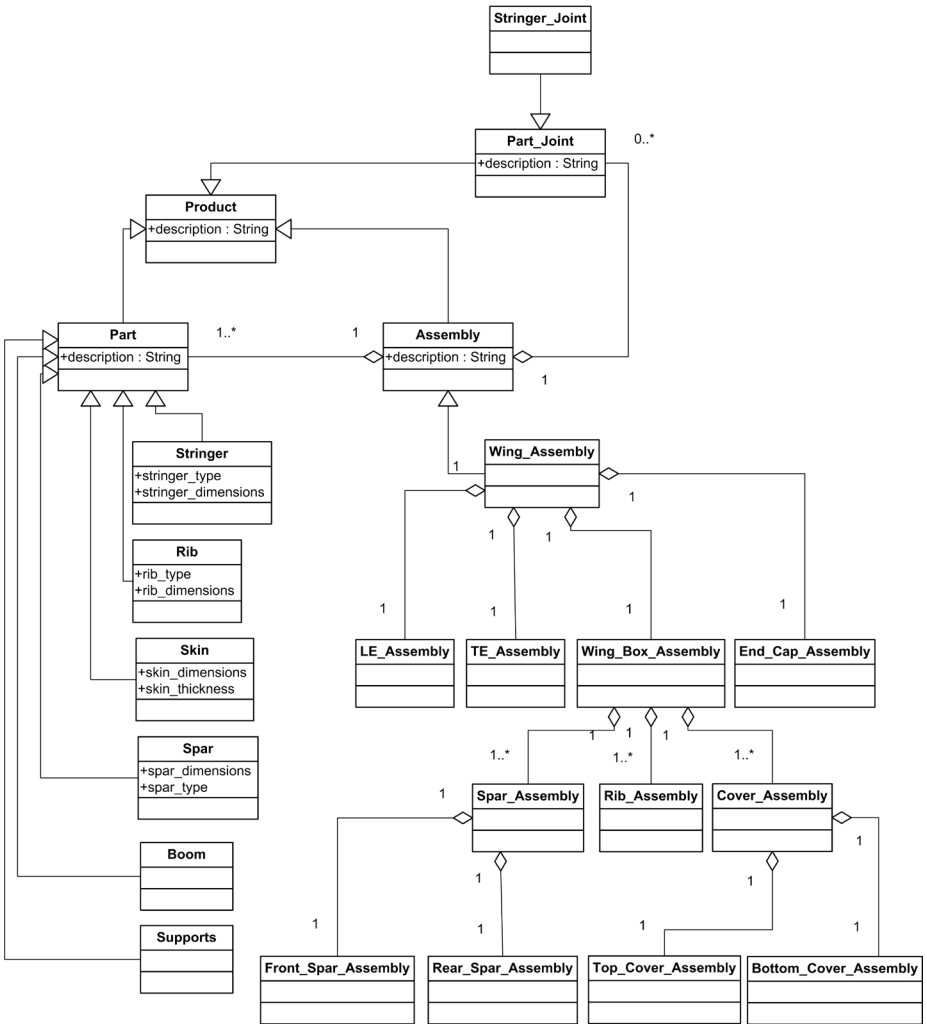

Figure 5.10: extended Product **class hierarchy for the manufacturing domain**

The domain-specific class hierarchy for the **Process** class is represented in Figure 5.11. For this domain-specific hierarchy, the focus is again on **Process** subclasses that can be used for structuring and annotating domain knowledge. Two subclasses are of particular interest: **Manufacturing_Process** and **Cost_Modelling_Process**. The former contains classes for the preparation, processing and post-processing steps that are also used to organise the cost model. Examples are given for each of these steps: **Preparation_Process** for instance contains a **Tool_Drying** class, **Processing** for instance contains **Automated_Tape_Laying**, and **Post_Processing** contains for instance **Non_Destructive_Testing**. To keep the figure clear, only two examples have been given per step; the domain ontology implemented in the solution has many more subclasses. The **Cost_Modelling_Process** class is inspired by the IDEF0 representation of the cost modelling task, as modelled in Section 5.2.2 (Figure 5.4). Its constituent activities are not included into the Figure.

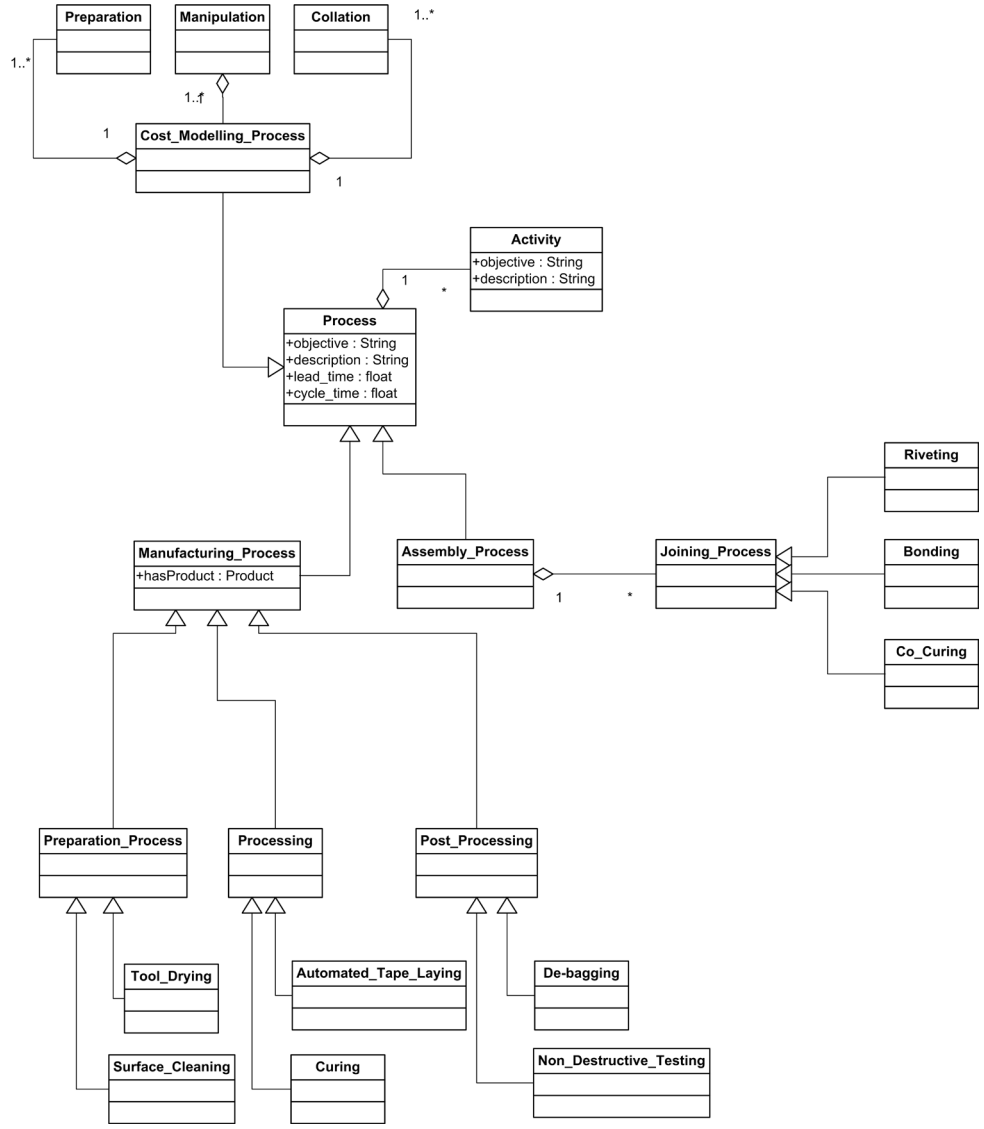

Figure 5.11: extended Process **class hierarchy for the manufacturing domain**

The **Resource** class hierarchy for the manufacturing domain is also largely similar to the design domain class hierarchy – see Figure 4.12. Some changes have been made to the **Material_Resource** class (Figure 5.12). An attribute specifying the fibre processing type (e.g. prepreg, dry fibre) has been added to the **Fibre_reinforcements** class. Similarly, an attribute specifying the matrix material type has been added to the **Matrix_Material** class. Furthermore, an **Equipment_Resource** class has been added to represent manufacturing equipment into the domain ontology.

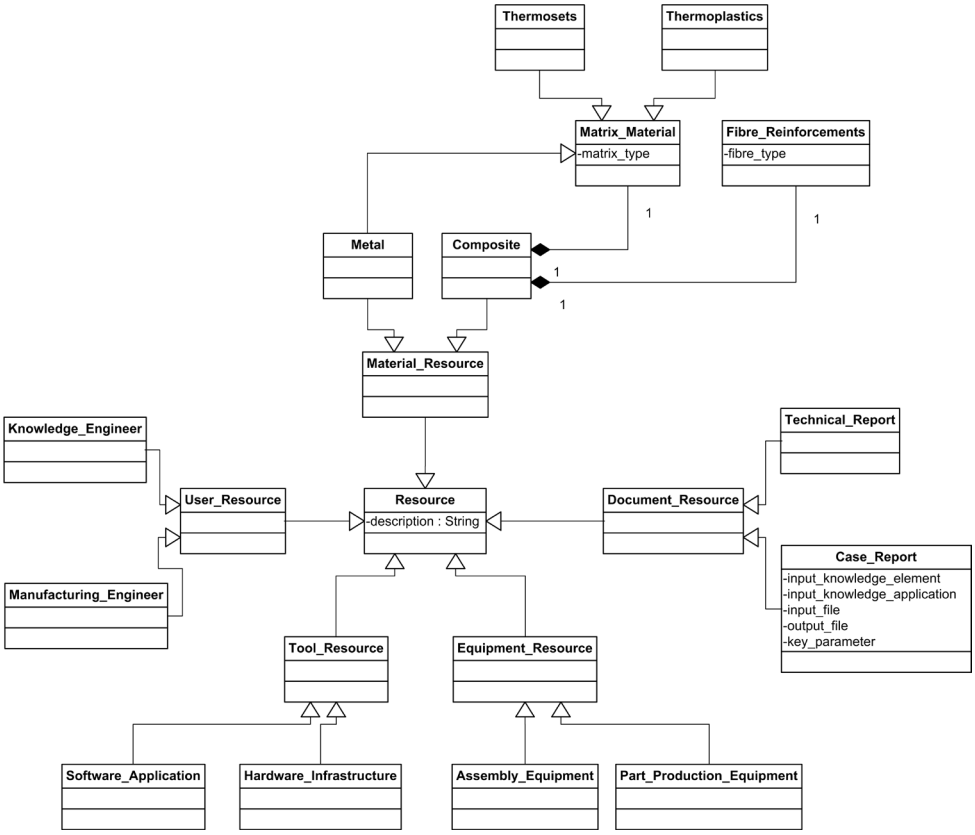

Figure 5.12: extended Resource **class hierarchy for the manufacturing domain**

The manufacturing domain ontology consists of the combined **Product**, **Process** and **Resource** class hierarchies and associated relations. It has been used to structure the captured knowledge and will be used in the subsequent step to annotate (elements of) the developed solution. This is further explained in the following Section.

5.3.4 Modelling & Implementation

This step consists of two activities: modelling of Enterprise Knowledge Resources (EKRs) for composite manufacturing cost modelling and estimation, and implementation of the EKRs into a functioning solution.

5.3.4.1 Solution Development: EKR Modelling

In modelling EKRs for the cost modelling and estimation problem, the focus is on a solution that enables knowledge change to be managed, while offering improved knowledge utilization and maintainability. This is indicated in Figure 5.13. This Figure expresses that modular cost model 'building blocks' can be stored in a

shared and managed knowledge base. From this knowledge base, users can assemble cost models using specific building blocks. Users can then make changes to the knowledge contained in the cost models: these changes are communicated to the knowledge base, where the changes can be incorporated after validation of correctness and reliability.

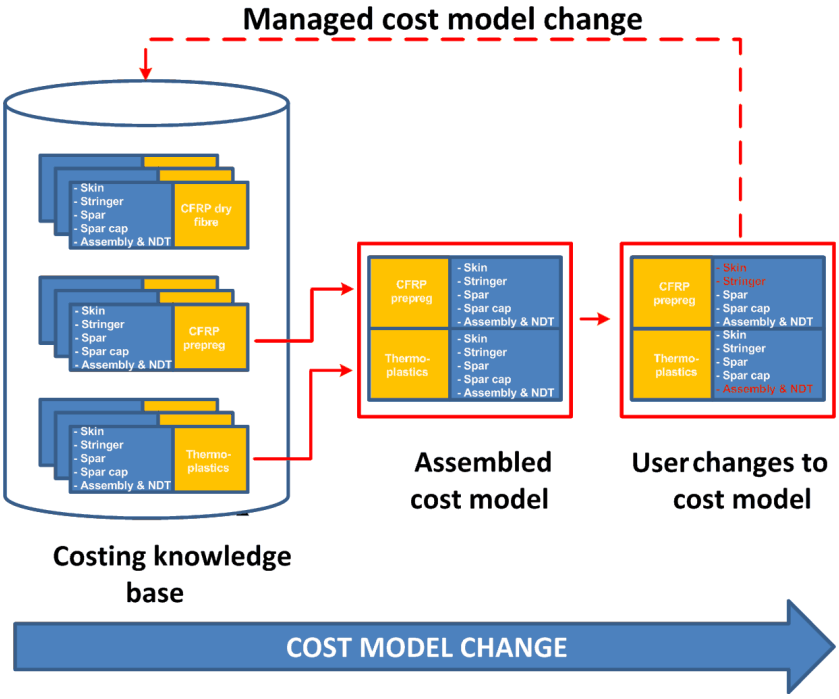

Figure 5.13: Managed cost model evolution

To realize the vision expressed in Figure 5.13, the Enterprise Knowledge Resource concept from the KLC ontology is employed. The following EKR classes have been modelled:

- **Enterprise Knowledge Resource**: A set of EKRs have been modelled to enable a modular approach for flexible assembly of unique cost models. To achieve this, the existing cost model has been reverse-engineered to define 'building blocks' that contain all the geometry inputs, process inputs, user inputs and model formulas belonging to a specific Product-Process-Resource combination. A single EKR formalizes a single building block and represents a single cost estimation task for manufacturing of a specific Product-Process-Resource combination. An EKR uses a semi-automated process to combine knowledge input parameters and formulas into a functioning, self-contained cost model in an Excel spreadsheet.

Several EKRs or building blocks can be put together to define a complete cost model. The use of building blocks in this manner has the desired effect of achieving central management of the inputs and formulas, instead of having multiple spreadsheet instances of the cost model.

- **EKR_Knowledge:** the EKR uses knowledge elements which are kept in individual knowledge articles. Some examples have been presented in Section 5.3.1. The knowledge articles capture the main cost model entities: process parameters, manual geometry parameters, user inputs and model equations. These elements are modelled and implemented on an individual basis to allow knowledge change to be managed in the knowledge base: if a user changes a single parameter in the spreadsheet cost model, this change can be back-propagated to the knowledge base, where only a single knowledge element is changed.
- **EKR_Process:** the EKR uses a process model for combining the individual knowledge elements (process parameters, user inputs, model equations and geometry – either from manual entry or imported from CATIA) and outputs them in a pre-configured Excel worksheet. For each EKR, the subprocesses and detailed steps/activities are modelled in the EKR. Consider the example from Section 5.3.1, T-stringer production (Table 5.4). The process model consists of the overall process, subprocesses and detailed processes as given on the left-hand side of Table 5.4; it is given in process form in Figure 5.14.

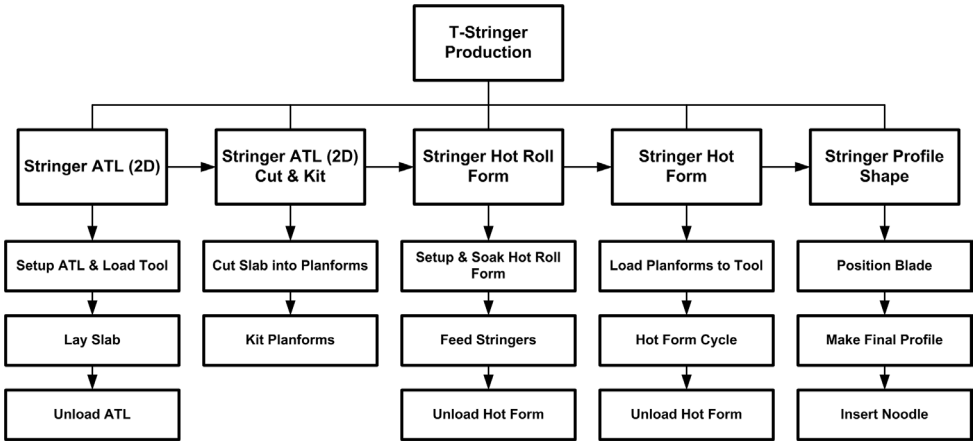

Figure 5.14: Process model for CFRP T-stringer production EKR

- **EKR_Case:** Spreadsheet results from generated cost models and subsequent user manipulations are not yet stored into a central case repository. Instead, parameter changes are tracked within the

spreadsheet while being used. Upon changes in the worksheet, parameter and/or formula changes are fed back into the knowledge base.

Using the preceding considerations, an EKR class diagram has been modelled for this specific case study and associated task. The UML class diagram is shown in Figure 5.15 and includes the **EKR_Knowledge**, **EKR_Process** and **EKR_Case** classes. Note in particular the **Knowledge_Element** class, which contains the specific attributes necessary for the manufacturing case study, including the excel_name, baseline_value and target_value attributes.

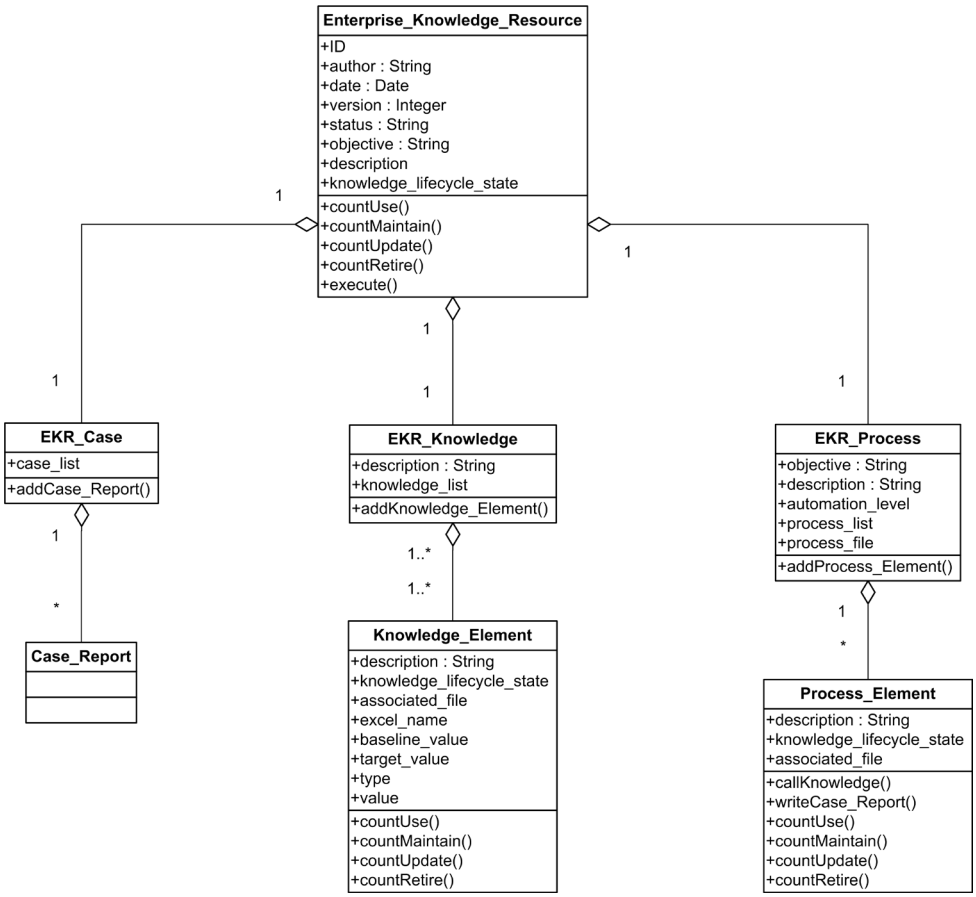

Figure 5.15: EKR class diagram (UML) for manufacturing case study

The EKRs and constituent classes such as **Knowledge_Element** are annotated using the previously introduced domain ontology. The composite manufacturing material resource, product breakdown structure and manufacturing process class hierarchies together offer the necessary classification richness to annotate all possible EKRs that can be used to compose a cost model for this case study. Using

these classification hierarchies allows for unique combinations of semantic tags for annotation of a specific building block, supporting search and retrieval by end users. An example of this is given in Figure 5.16, where PPR classes (**Equipment_Resource**, **Material_Resource**, **Manufacturing_Process**, **Part**, **Assembly**) taken from the domain ontology class hierarchies as given in Section 5.3.3 are associated with the **Enterprise_Knowledge_Resource** class. Similarly, the other EKR classes (such as **Knowledge_Element** and **Process_Element**) can be annotated using the same PPR annotation tags. A specific annotation example is given in the next Section.

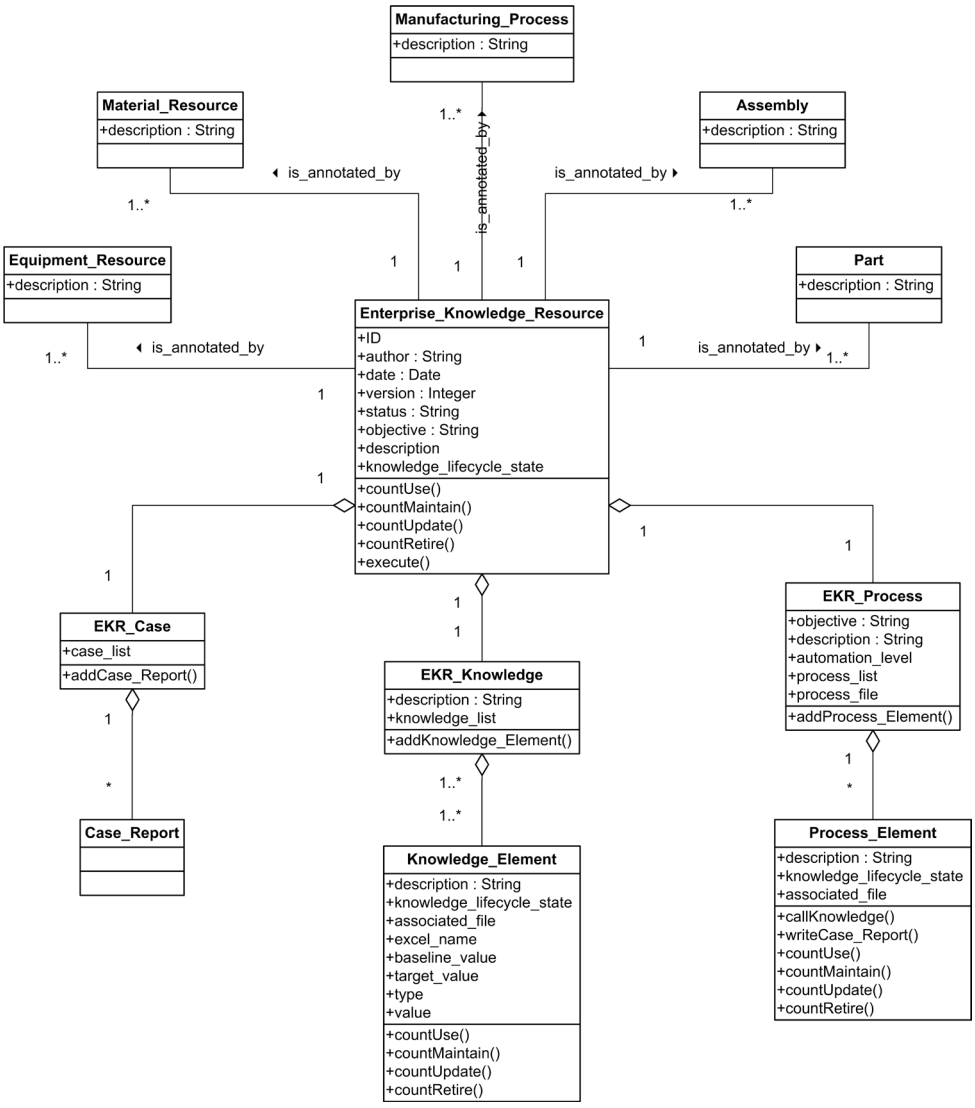

Figure 5.16: Semantic annotation of a cost model EKR

The EKRs can be used to construct an integrated cost model. The proposed solution must be able to manage duplicated knowledge elements when multiple EKRs with similar knowledge inputs are selected and exported. Also, the solution must have the capability of coping with changes in the spreadsheet environment and feeding back these changes into the knowledge base. This process must be subjected to a validation process: updates in the knowledge base must be shared and agreed upon before acceptance of knowledge base changes. These issues are addressed in the EKR implementation section of the solution development.

5.3.4.2 Solution Development: EKR Implementation

To implement the EKR / building block approach and associated models, an implementation architecture has been devised (Figure 5.17).

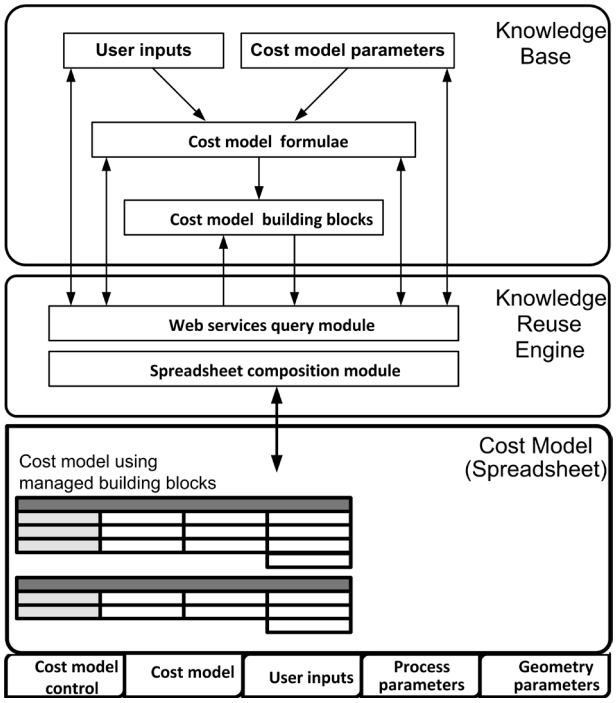

Figure 5.17: Implementation architecture

The architecture consists of the following main elements:

- **Knowledge Base:** this repository holds the EKRs (cost model building blocks), including the knowledge elements and process models that are

used to compose the cost model building blocks. The knowledge base has been built in Ardans Knowledge Maker (AKM). Each EKR element is represented by a knowledge base instantiation, i.e., an AKM knowledge article. An example of a knowledge element implemented in AKM has been given in Figure 5.9. When the knowledge in an article is updated, a new version is made. The old version is stored, but the new version becomes the 'default' representation and is used in cost model composition.

- **Knowledge Reuse Engine:** this architectural element drives the querying of the knowledge base and retrieval of knowledge elements for the cost model building blocks. It consists of a web services query module and a spreadsheet composition module. The latter has been composed using a VBA macro in Microsoft Excel.
- **Excel Spreadsheet:** the spreadsheet application receives the building blocks from the web service. The spreadsheet composition module automatically builds the resultant cost model, with all inputs, formulas and outputs in place. Any double entries are checked and only one representation is maintained within the model. The cost model inherits default values from the knowledge base for the user inputs, but these and the other parameters can be changed to suit the user needs. A function is included to compare the cost model elements with the knowledge contained in the knowledge base; if the user makes changes in the open spreadsheet, they can then compare the resulting changes with the original values and subsequently choose to update the knowledge base based on the outcome. Minor and major changes to the cost model (e.g. changes in formulas or changes in building blocks) can be performed by updating the relevant knowledge articles. Through the modularised approach, the changes are automatically incorporated into the building blocks and subsequently into any generated spreadsheet-based cost model.

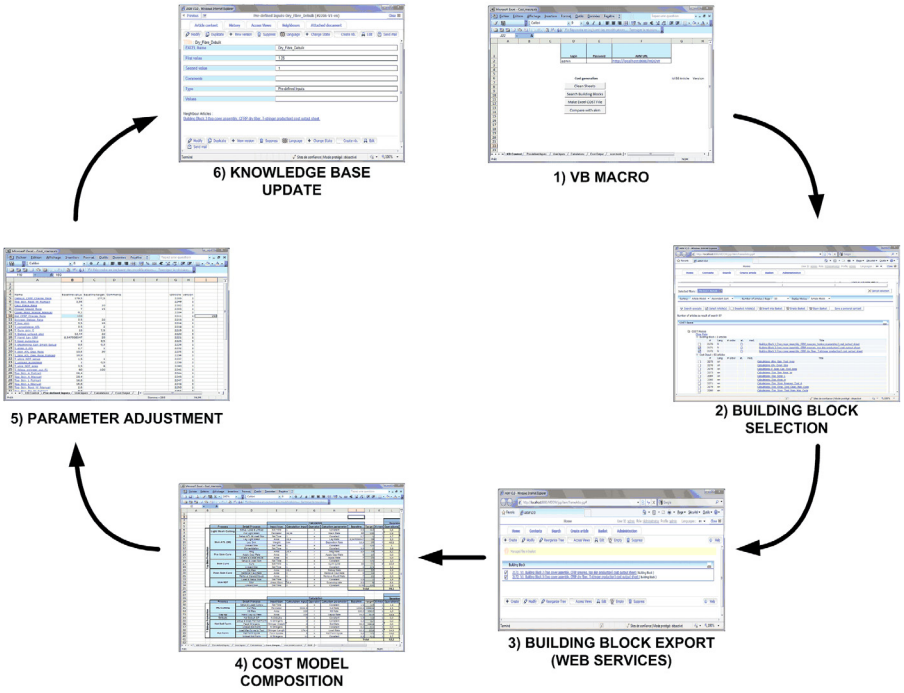

Figure 5.18: User process for cost model composition using proof-of-concept solution

The user process for using the knowledge-based solution is given in Figure 5.18. To illustrate how the process works in practice, an example is presented here that was used as part of a validation exercise. The example concerns the production of CFRP T-stringers for a wing top cover, as presented before in Section 5.3.1. This EKR has been annotated using the classes shown in Figure 5.19. These are part of the manufacturing domain ontology introduced in Section 5.3.3.

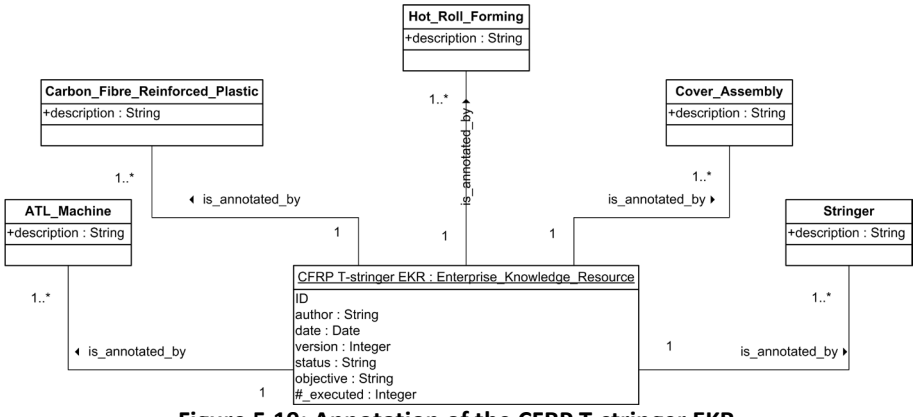

Figure 5.19: Annotation of the CFRP T-stringer EKR

The user now follows the following process to produce a cost model for this engineering task, as illustrated in Figure 5.18. The Figure shows that after opening a spreadsheet-integrated Visual Basic macro (step 1) and calling the AKM environment, the user employs the semantic labels as shown in Figure 5.19 to search for the EKR. The corresponding search result displays the suitable EKR(s) (step 2). The relevant EKR is subsequently selected (step 3) and exported to the spreadsheet application using a web service. The supporting Visual Basic macro automatically generates the cost model (step 4) that belongs to the building block(s). After this, the user is free to make choices and changes in the spreadsheet environment to arrive at customised process time and cost estimates (step 5). The changes can be checked against the existing knowledge base as a 'live link' is maintained between the two applications. If necessary, the knowledge base can be updated to support data, information or knowledge change (step 6).

5.3.5 Analysis & Delivery

The Knowledge Lifecycle Model, the KLC ontology and the KNOMAD methodology have been applied to construct a proof-of-concept solution for knowledge-based cost modelling and estimation of composite wing cover manufacturing processes. The solution meets the following case study requirements:

- It supports end users in the composition, use and control of cost models: through the EKRs, cost models can be composed. They can be used within a spreadsheet environment. Storing and managing the EKRs in a knowledge base means that the knowledge and process elements that make up an EKR are controlled: any changes can be validated and shared using AKM's functionality.
- The solution allows for flexible and fast estimation of cost for composite wing covers using the EKRs.

The implemented proof-of-concept solution consists of three independent EKRs – with respect to the full cost model, the functionality is limited. The functionality has been validated through successful application in practice, for which a representative example has been presented in the preceding Sections. The associated user process has been validated at the OEM company through expert opinion and user feedback. The involved experts have intimated that efforts related to cost model composition and maintenance would be reduced significantly when using the developed solution. However, as the functional solution has been developed to a proof-of-concept stage (at Technology Readiness Level 3, or TRL3), quantification of costs and benefits of the tool was not considered. The following qualitative observations can however be made.

Firstly, the solution to the case study provides knowledge life-cycle management through the inherent capabilities of the AKM tool and through the EKR approach. In particular, the provisions for explicating the justification behind knowledge elements, the 'live link' between the knowledge base and the spreadsheet application, and the possibility to track the change of knowledge through the retention of historical knowledge articles contribute to a significant improvement with respect to the legacy cost model.

Secondly, the solution also facilitates knowledge utilization through the retention and exploitation of existing and legacy models, i.e., spreadsheet models, which has ensured that users keep using familiar and trusted tools and processes. Furthermore, the use of a dedicated knowledge base, with the associated provisions to ensure the availability of trustworthy knowledge, assures users that the right knowledge is available at the right time.

Thirdly, the solution addressed knowledge transparency by including semantic annotation and provision for knowledge explication. The difficulty with actually understanding complicated or dense legacy tools has been highlighted so that visibility is interpreted primarily in terms of the user being able to understand the rationale embedded into the tool.

In its current proof-of-concept incarnation, the knowledge-based cost modelling solution is limited in several aspects. First, the whole process is only semi-automatic. Composition of a cost model still requires non-productive steps that do not add value, though the overall time spent on model composition is reduced. Another limitation is the fact that the spreadsheet results from generated cost models and subsequent user manipulations are not yet stored into a central repository for the company, as indicated in Section 5.3.4.2. Implementing this would amount to better adherence to the 'Case' part of the EKR approach. A final limitation of the research is that the overall cost modelling capacity has not yet been fully addressed in informal terms: whereas the individual knowledge articles do feature informal knowledge, the higher-level process does not.

5.4 Discussion of Results

With respect to the usability and maintainability challenges that are associated with the contributions of this dissertation, the following can be observed:

- **Moving beyond black-box applications and ensuring transparency:** The developed solution is designed to improve upon the current black-box implementation of the cost modelling capability. To achieve this, the solution does not replace the natural use of spreadsheets to compute the cost of components. However, it manages the knowledge driving the cost model and deploys it to a working spreadsheet from which users can

understand the rationale of the computed cost. Knowledge is categorized, easily accessible and usable, and can be maintained and/or updated over life. Previous versions of the cost model (elements) are stored. Furthermore, the use of the EKR approach makes the cost model solution transparent. A cost modelling and estimation EKR makes it clear which knowledge is involved within the cost modelling and estimation solution, which inputs are necessary, which steps are taken within the process and which outputs are generated. The knowledge base enables storing, justifying and updating cost model knowledge elements and records previous versions of the cost model.

- **Task orientation:** The EKRs represent specific engineering tasks in the form of modular cost modelling elements. They offer the possibility to compose a cost model using modular building blocks. The resulting tailor-made cost model can be used to estimate cost. It can also be configured according to user choice, allowing cost performance investigation for competing solutions.
- **Expert / end user involvement:** The solution uses the EKR approach, but does not replace the use of spreadsheets to compute the cost of components. It manages the knowledge driving the cost model and deploys it to a spreadsheet which estimates costs. Users are already familiar with the spreadsheet environment; its continued use may be an important factor in user adoption of the solution. Through the spreadsheet environment, users can understand the rationale of the computed cost and access the underlying knowledge.

6 Maintenance Case Study: Supporting Wing Maintenance – B737 Leading Edge Slat Downstop Assembly Modification & Inspection

This chapter describes the development of a knowledge-based application that supports the digitalization and execution of maintenance tasks. As such, this case study focuses on the maintenance domain of the aircraft lifecycle. Together, the case studies will shed light on how the overall research objective can be achieved, with emphasis on the latter part of the objective: "Support consistent formalization, use and maintenance of changing knowledge within aircraft lifecycle phases *to improve domain-specific modelling, execution and control of engineering tasks*". The case studies also offer a practical perspective on the following research questions:

- How can knowledge change be accommodated during KBS development?
    - Which models are required and how do these models help to accommodate knowledge change?

This case study presents an approach to support maintenance task digitalization and execution for wing components. First, the case study context and challenges are introduced. Subsequently, the theoretical contributions are applied to the maintenance domain: the Knowledge Lifecycle model is used to qualify *and* quantify knowledge change, task analysis is performed to prepare the use of the KLC ontology and the KNOMAD methodology is applied to the case, resulting in a flow chart for the development of a knowledge-based solution. As in the previous case study, the solution supports the deployment and use of knowledge as an element in modular knowledge packages (the previously introduced Enterprise Knowledge Resources) that are managed in a central knowledge repository. An EKR can be deployed to support a maintenance task. Development and implementation of the solution are discussed in detail in Section 6.3. Validation of performance with respect to the case study objective(s) and requirements is briefly indicated in Section 6.3.5: Analysis & Delivery. The case study concludes with a discussion of the results within the context of the dissertation objectives and contributions to theory.

6.1 Case Study Context and Challenges

Literature for the aircraft MRO domain tends to focus on performance measurement and optimization of maintenance processes such as planning, management and execution (Garg and Deshmukh, 2006) and on the relation between maintenance and safety (Wartan, 2010). In marked contrast to the

design and manufacturing domains, literature regarding the development and use of advanced information technology (IT) such as PLM systems or knowledge-based (engineering) applications is very limited. This is supported by Lee *et al.* (2008), who note the low adoption of PLM technology in maintenance – see Figure 6.1.

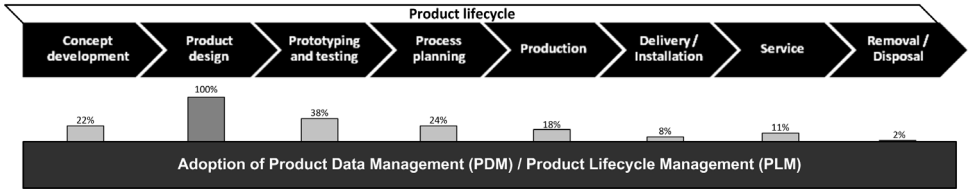

Figure 6.1: Adoption of PLM in the MRO domain (adapted from Lee *et al.* (2008))

Lee *et al.* (2008) identify a number of requirements for the adoption of knowledge systems within the aircraft MRO domain. In particular, automated information retrieval, associative inspection and maintenance procedures and tools, product structure information, and fault detection & isolation tools should be provided. Ideally, a PLM application or knowledge-based application should be able to record, check and manage inspection and maintenance records. Currently, processes are independent and largely manual (Lee *et al.*, 2008). Furthermore, the adoption and impact of information systems within maintenance is low; it is also an area of relatively limited research interest (Garg and Deshmukh, 2006).

This is an indication of a larger problem – the relatively low level of digitalization in the maintenance domain, and the reliance on legacy, paper-based approaches to execute and record maintenance tasks. Findings from TU Delft research performed at a large Dutch aircraft maintenance provider show that a paper-based approach is prevalent in carrying out maintenance work; processes, work instructions, safety instructions and maintenance reports are predominantly kept on paper (Wartan, 2010). The resulting records are stored in archives for the purposes of demonstrating airworthiness compliance during aircraft phase-out (Burhani, 2012). Similar findings are reported by Lampe *et al.* (2004), who point out the labour intensive manual documentation and check procedures at MRO providers. Lampe *et al.* (2004) indicate that the time associated with searching for appropriate documentation can amount up to 15-20% of the total work time of a mechanic.

Over the last years, the situation has improved as digital tools for providing maintenance information and supporting maintenance tasks have been developed and put to use. A fairly representative illustration of current practice is provided by Lampe *et al.* (2004) – see Figure 6.2. This figure highlights that current maintenance work is supported by a mixture of paper and digital documentation as well as tools, materials and parts required for the job.

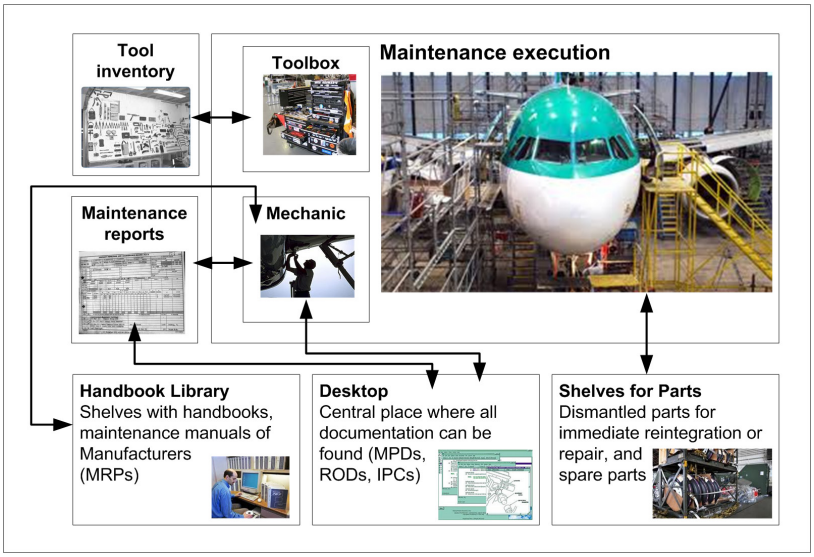

Figure 6.2: The aircraft MRO environment (adapted from Lampe *et al.* (2004))

Within MRO processes, an increasing number of supporting documentation is offered digitally. This includes OEM documentation such as the Airplane Maintenance Manual (AMM), Maintenance Planning Document (MPD), Illustrated Parts Catalogue (IPC), Structural Repair Manual (SRM) and Service Bulletins (SB). These documents can be offered through the OEM's web portal (e.g. Baker *et al.* (2006)) or as part of OEM software (Airbus, 2012).

However, despite recent advances, a number of major issues still remain:

- **Legacy work processes & systems:** remaining aspects of paper-based approach lead workers to shortcut the process as it takes too long to collect the relevant documents: safety and efficiency are compromised (Wartan, 2010).
- **Information exchange across stakeholders:** various stakeholders hold different information necessary for the successful execution and record keeping of maintenance tasks. For instance, MyBoeingFleet.com can provide the OEM information, the FAA or EASA holds the regulatory information (Airworthiness Directives or ADs), the airliner holds engineering orders (EO) and maintenance records. This information needs to be exchanged and be available to the end user in an integral way.
- **Maintenance report keeping and data accuracy:** some proof-of-concept research regarding the use of RFID tags to support automatic maintenance documentation has been performed (Lampe *et al.*, 2004). However, recent findings (Burhani, 2012) suggest that report keeping is still a manual job that

has only partly transferred into digital format. The manual entry of maintenance data is error-prone and may cause issues with data accuracy and completeness.

It is important to address these issues as a structured approach to data, information and knowledge capture, storage and use in MRO organizations has implications for data-driven research and improvement (Jagtap and Johnson, 2011) from a maintenance domain perspective as well as a through-life perspective. For the maintenance domain, capturing and storing in-service information such as component life times, types of failures, rate of failures, cost of spares, lead-time of spares, amount of non-routine job cards, etc. may be used to evaluate and predict product reliability, availability and maintainability, and consequently help optimize maintenance processes and planning. From a through-life perspective, the incorporation of knowledge about the in-service performance of existing products can lead to improvements in the design of new products (Jagtap and Johnson, 2011).

To summarize, the broad research problem for the maintenance domain is that the paper-based approach to aircraft maintenance is insufficient, as it has an adverse effect on process efficiency and safety. Though digital tools for maintenance support are increasingly being used, data capture and information exchange is complicated. Within MRO providers, legacy systems and insufficient means for maintenance data capture, storage and quality control are a problem. Between OEM, operators and regulators, systems are not integrated.

To resolve these issues, a potential direction is to move towards a 'push-of-the-button' digital solution for capturing and using aircraft maintenance task knowledge, processes and history to support maintenance execution and prove continued airworthiness compliance.

To narrow the scope associated with the research problem, this case study considers a particular maintenance task related to wing maintenance for the Boeing B737: modification and detailed inspection of the main track downstop assembly of the leading edge slats (Boeing, 2010). This task is associated with an FAA Airworthiness Directive (FAA, 2007) and a revised Service Bulletin (SB) issued by the OEM, Boeing (Boeing, 2010).

The consolidated case study objective is to construct a proof-of-concept solution for capturing and using aircraft maintenance task knowledge, processes and history relative to the modification and detailed inspection of the B737 main track downstop assemblies of the leading edge slats. The following requirements must be met:

1) The solution must support end users in execution of the maintenance task;
2) The solution must offer a digital means of record keeping in order to prove continued airworthiness compliance;
3) The solution must be automated to the fullest extent possible.

Validation with respect to these requirements is performed in Section 6.3.5: Analysis & Delivery.

The introduced problem is related to knowledge change and consequently to research challenges regarding knowledge usability and maintainability. The following issues will be taken into account in the case study:

- **Moving beyond black-box applications and ensuring transparency:** as it dispersed within and across organisations, maintenance knowledge (e.g. ADs, SBs, maintenance reports) is difficult to access, inspect and maintain. To support knowledge maintenance, it is necessary to support categorization, accessibility, traceability and subsequent sourcing of knowledge. The solution must enable the storing, justifying and updating of maintenance knowledge elements and processes and must support maintenance record keeping. To ensure transparency, it should be clear which knowledge is involved for specific maintenance tasks: which inputs are necessary, which steps are taken within a process and which reports are generated with what kind of data? The solution must enable a standard approach of supporting maintenance tasks.
- **Task orientation:** knowledge involves a 'capability for effective action'. The capability for action can be met by explicitly associating sets of knowledge with functional tasks, i.e. performing specific maintenance tasks. The solution will offer support for maintenance task execution by offering a 'one-stop' portal for the documentation, process models and maintenance reports associated with that task.
- **Expert / end user involvement:** End users must be able to identify, use, interact with and if necessary, maintain or update the data, information and knowledge related to a specific maintenance task.

Through these considerations, the case study contributes to validation of the overall research contributions to theory. This is discussed in Section 5.4: Discussion of Results.

6.2 Application of Theory to Maintenance Case Study

Before developing a solution, this section acts as an intermediate step by applying the developed theory to the case. First, the Knowledge Lifecycle Model is applied

to identify knowledge change for the maintenance task. This task is subsequently analysed in support of further application of the KLC ontology to solution development in the Results section. Finally, it will be shown how the KNOMAD steps will be applied to this case to guide the subsequent solution development in Section 4.3: Results.

6.2.1 Application of Knowledge Lifecycle Model: Identifying Knowledge Change

For this use case, change of knowledge specific to the main track downstop assembly modification and inspection task is analysed. This is done using two related, formalized representations of this knowledge: Airworthiness Directives and Service Bulletins.

Airworthiness Directives (ADs) are regulatory documents that are issued against certified aeronautical products with the purpose of notifying aircraft owners of 'unsafe conditions, non-conformity with the basis of certification and other conditions affecting the airworthiness of their aircraft' and 'the mandatory actions required for the continued safe operation of an aeronautical product' (Transport Canada, 2002). An AD is typically created in the days, weeks or even months after a (potentially) unsafe condition has been discovered. Following its effective date, an AD must be complied with by aircraft owners to maintain airworthiness.

Service Bulletins are documents that are issued by the Original Equipment Manufacturer (OEM) to either recommend actions for the improvement of an aircraft or to support corrective actions, typically in relation with an associated AD.

- **Airworthiness Directives:** four ADs have been issued in relation to the main track downstop assembly modification and inspection task. The first issue was Emergency AD 2007-18-51. This Emergency AD describes the unsafe condition, effective date, applicability, required actions and compliance times for this specific maintenance issue.
    - **Unsafe condition:** the Emergency AD was issued following reports of parts coming off the main slat track downstop assemblies. Following slat retraction, the slat track housing was punctured in two cases, leading to fuel leaking in one case, and fuel leaking and a subsequent catastrophic fire (aircraft loss) in another case.
    - **Effective date:** the effective date for Emergency AD 2007-18-51 was 25 August 2007.
    - **Applicability:** The Emergency AD applied to all Boeing Model 737-600, -700, -700C, -800, -900, and -900ER series airplanes, certificated in any category.

- **Required actions & compliance times:** Emergency AD 2007-18-51 required a repetitive detailed inspection of the main slat track downstop assemblies for verification of proper installation of the slat track hardware (bolt, washers, downstops, stop location, and nut) and a one-time torquing of the nut, both within 24 days after reception of the AD. During inspection, if any part were to be missing or installed incorrectly, replacement was required as well as a detailed inspection of the inside of the assemblies for foreign object debris and damage. Removal of any debris and repair of any damage was required. The whole detailed inspection procedure was required to repeat at intervals not exceeding 3.000 flight cycles.

Following the detection and reporting of additional parts coming off the main track downstop assemblies, emergency AD 2007-18-51 has been superseded by Emergency AD 2007-18-52, effective date 28-08-2007. This AD has been issued to change the compliance time for the detailed inspection of the assemblies from 24 days to 10 days after receipt of the Emergency AD. In addition, Emergency AD 2007-18-52 determines that a borescope inspection of the assemblies is acceptable in lieu of detailed inspection. Emergency AD 2007-18-52 has been subsequently formalized into AD 2007-18-52, effective date 26-09-2007. This AD serves as a formal entry into the Federal Register, instead of the preceding emergency AD, which acted as a temporary measure. These subsequent issues can be characterised as 'maintain' and 'update' knowledge actions, respectively.

Finally, AD 2007-18-52 has been superseded by AD 2011-06-05, effective date 26-04-2011. The new AD requires the replacement of downstop assembly hardware with new hardware while keeping the previous requirements with respect to performing inspections of the slat cans on each wing and the lower rail of the slat main tracks for debris, replacing the bolts of the aft side guide with new bolts, and removing any debris found in the slat can. Furthermore, AD 2011-06-05 removes some B737 models from the applicability. As this new AD changes both the context and content of knowledge, it can be characterised as an 'update' action.

- **Service Bulletin:** two Service Bulletin versions have been issued by the OEM, Boeing, to address the modification and detailed inspection of the main track down stop of the leading edge slats.

Boeing Service Bulletin 737-57A1302, dated 15 December 2008, describes the planning information, material information and accomplishment instructions necessary for the removal of existing downstop assemblies and installation of new ones, as well as removal of

existing aft side guide attach bolts and replacement with drilled head bolts lock-wired together on the ribs of the wing leading edge (Boeing, 2010). It also describes the inspection process. As such, this SB is a response to the requirements of AD 2007-18-52 as well as precursory documentation for AD 2011-06-05, which actually mandates the replacement of downstop assembly hardware.

Boeing Service Bulletin 737-57A1302, Revision 1, dated 18 October 2010, changes and adds upon the previous SB by adding a borescope[††] inspection option for the slat main track, by adding part and tooling information, by changing a tolerance limit for a certain repair option, by introducing or updating process step information, and by removing an aircraft from the applicability. This SB can therefore be characterised as an 'update' knowledge action.

Based on the changes in these ADs and SBs, one can state that the knowledge required for the use case maintenance task is subject to change. For the Airworthiness Directives, both the context and content of the knowledge encapsulated within the ADs is subject to change: applicability and compliance times (context) are changed in AD 2007-18-52 and AD 2011-06-05, whereas an additional inspection and assembly replacement (context) are mandated in AD 2011-06-05. For the SBs, Revision 1 of Boeing Service Bulletin 737-57A1302 changes context (applicability, part information) and content (process step descriptions, tolerance limits, inspection option).

Similar to the previous case studies, the preceding discussion qualitatively shows that knowledge is subject to change with respect to the maintenance task (modification and detailed inspection of the main track downstop of the leading edge slats of most Boeing B737 types).

6.2.2 Application of Knowledge Lifecycle Model: Quantifying Knowledge Change

In the following Section, the Knowledge Lifecycle Model concepts will be tested using data from the maintenance stage of the aircraft lifecycle. Specifically, the idea of using knowledge actions will be used to quantify knowledge change in Airworthiness Directives (ADs). In the following sections, the research data, hypotheses and sample will be described, followed by analysis.

[††] An optical device consisting of a rigid or flexible tube with an eyepiece on one end, an objective lens on the other end, linked together by a relay optical system.

6.2.2.1 Description and Operationalization of Research Data: Airworthiness Directives

To quantify changes in knowledge, a sample of Airworthiness Directives has been gathered. As mentioned, Airworthiness Directives (ADs) are regulatory documents that are issued against certified aeronautical products with the purpose of notifying aircraft owners of 'unsafe conditions, non-conformity with the basis of certification and other conditions affecting the airworthiness of their aircraft' and 'the mandatory actions required for the continued safe operation of an aeronautical product' (Transport Canada, 2002). More importantly, they are also a highly formalized form of knowledge, as they consist of knowledge context in the form of metadata such as AD date, type approval holder & type/model designations, applicability and effective date, etcetera), knowledge content (related technical details (sometimes expressed in associated Service Bulletins (SBs) provided by the Original Equipment Manufacturer (OEM), and provisions to maintain compliance) and a capability for effective action (required actions and compliance times) – see also Transport Canada (1996) and AD examples retrievable from the European Aviation Safety Agency (EASA, 2012). Furthermore, they are rigidly maintained in online regulatory knowledge bases (EASA, 2012) due to their critical role in ensuring aviation safety. In effect, ADS are knowledge elements – and they undergo changes throughout their life. An AD is typically created in the days, weeks or even months after a (potentially) unsafe condition has been discovered and is in effect from its effective date. From that point on, an AD must be complied with by aircraft owners to maintain airworthiness. Sometimes, ADs are revised to reflect changes in their knowledge context, for instance with respect to applicability across aircraft types. ADs can also be superseded, in which the content of an AD is changed, while frequently the context is as well. Finally, ADs can be cancelled, for instance when the aircraft type(s) for which the AD was issued are not operated anymore. Following this description, ADs can be mapped onto the Knowledge Lifecycle Model by operationalization of the knowledge actions in terms of the actions that are taken with respect to ADs:

Create ↔ an original AD is created.

Use ↔ an AD is in effect and is used by aircraft owners. This knowledge action is *not quantified in this case study* as it has proven impossible to track the number of use actions by aircraft owners relative to specific ADs.

Maintain ↔ an AD is revised (update of context element of AD knowledge, for instance applicability to aircraft type).

Update ↔ an AD is superseded (update of both context and content of AD knowledge).

Retire ↔ an AD is cancelled.

6.2.2.2 Case-Specific Hypotheses

Based on the Knowledge Lifecycle Model and the selection of ADs as the sample population, a number of case-specific hypotheses can be posited. The first hypothesis concerns the behaviour of knowledge, as operationalized in the knowledge actions create, maintain, update and retire.

H_1: The frequency of knowledge actions decreases along the knowledge lifecycle

This hypothesis is motivated by the idea that most knowledge elements would not move beyond creation and use, i.e. the actions maintain and/or update and/or retire would not be applied to most knowledge elements.

In contrast to the first hypothesis, which is concerned with the behaviour of knowledge throughout its life, the following set of hypotheses is related to the behaviour of knowledge over aircraft life.

H_2: Number of knowledge actions per year increases during the aircraft lifecycle
H_3: Use of the 'create' action per year increases during the aircraft lifecycle
H_4: Knowledge change per year increases during the aircraft lifecycle
H_5: Use of the 'maintain' action per year increases during the aircraft lifecycle
H_6: Use of the 'update' action per year increases during the aircraft lifecycle

Hypothesis H_2 is motivated by the idea that knowledge about maintenance in the form of unsafe conditions to be resolved increases during the aircraft lifecycle, and therefore the number of associated knowledge elements will increase. Hypothesis H_3 is similar to the previous one, but instead of considering the total number of knowledge elements, the number of 'create' actions is considered.

Hypothesis H_4 is motivated by the cumulative effect expressed in the previous hypotheses, and particularly H_1: as the yearly number of knowledge elements increases, the total number of knowledge elements (integrated over time) will be increasingly large. I.e., should hypothesis H_1 be true, the total number of knowledge elements does not grow at a constant rate but at an increasing rate. Should knowledge change per year be present at a certain constant rate, the logical result is hypothesis H_4: an increase of knowledge change during the aircraft lifecycle. The remaining two hypotheses are a segregation of the preceding hypothesis in terms of the individual knowledge actions: instead of aggregate knowledge change, the actions 'maintain' and 'update' per year are hypothesized to increase over time.

6.2.2.3 Description of Research Sample

For this initial attempt at quantifying knowledge change, two samples of ADs have been taken from the EASA AD Publisher Tool (EASA, 2012) and the FAA Regulatory and Guidance Library (FAA, 2012).

With respect to the EASA ADs, to limit the scope of the problem and enable reproducibility, only ADs with respect to the Airbus A320 aircraft have been taken from the Publisher Tool. The applied filter was 'Return all ADs from 1988-08-01 until 2011-09-27, applicable to the type A320 from TC holder AIRBUS'. The full AD history for the Airbus A320, starting from first flight deployment in 1988 and going to 27 September 2011 (sample final access date) has been taken to compile the initial sample, which consisted of 288 publications. This sample has been inspected and revised to account for AD revisions: in the initial sample, a revised AD (e.g. AD 2007-0065R2) would count as one publication, whereas in the revised sample, the revised AD counts as three publications – the originally created AD (AD 2007-0065) and its two revisions (AD 2007-0065R2, AD 2007-0065R1). The final revised sample consists of 418 publications.

With respect to the FAA ADs, a similar sample was chosen. Only ADs with respect to the Boeing B737 aircraft have been taken into consideration. This includes all subtypes of the B737 aircraft (-100, -200, -200C, -300, -400, -500, -600, -700, -700C, -800, -900, -900ER.). The full AD history for the B737 has been compiled, starting from 1968 and ending at 2012-03-20 (sample final access date). The final sample consisted of 493 original publications. As with the A320, the sample has been revised to account for AD revisions and supersedures. The final revised sample consists of 648 publications.

6.2.2.4 Results

The following two subsections describe the analysis of the A320 and B737 samples.

A320 Analysis

To analyse the knowledge actions along the knowledge lifecycle, the sample of 418 A320 AD publications has been analysed using a simple frequency count for the number of knowledge actions per category. The results are shown in Figure 6.3. Of the 418 actions associated with the publications, 240 were 'create' actions, 124 'maintain' actions, 47 'update' actions and 7 'retire' actions. Evidently, the number of knowledge actions decreases along the knowledge lifecycle, but a sizeable part of the A320 ADs are subject to knowledge change.

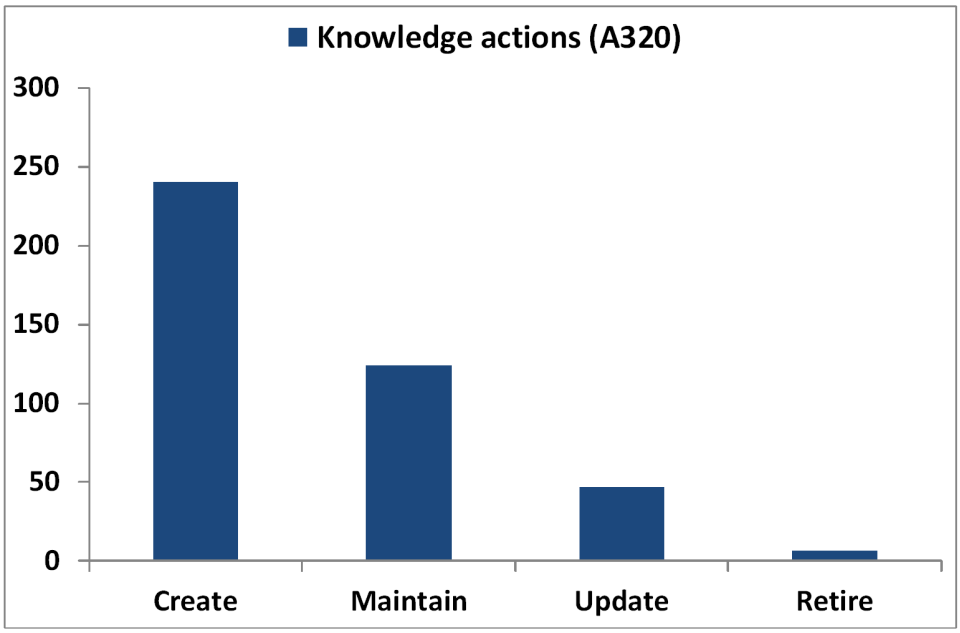

Figure 6.3: Frequency of knowledge actions (A320)

To analyse the yearly number of knowledge actions along the aircraft lifecycle, the combined number of knowledge actions per year for the sample period 1988 – 2011 has been enumerated, followed by a linear bivariate correlation analysis with the total number of knowledge actions per year as dependent variable and time (in years) as the independent variable. The results are given in Figure 6.4 and Table 6.1. From the table it can be seen that there is a significant ($p < .05$) but small relation ($R = .406$, $R^2 = .165$) between the total number of knowledge actions and the aircraft lifecycle. The effect size is heavily influenced by the variation over the years, especially visible in the dip for knowledge actions between 2007-2010.

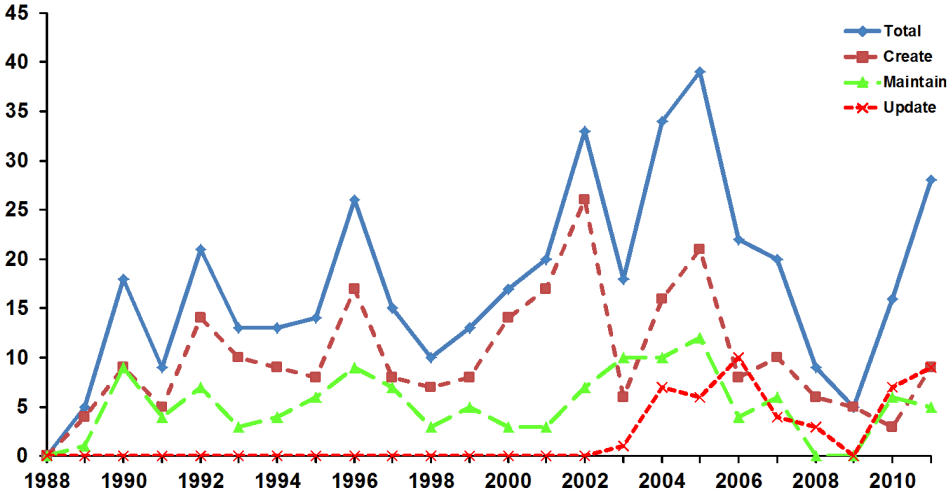

Figure 6.4: Knowledge actions per year versus the A320 lifetime (years)

Table 6.1: Bivariate correlation - knowledge actions per year versus lifetime (A320)

Correlations

		Number of knowledge actions per year	A320 Lifetime
Number of knowledge actions per year	Pearson Correlation	1	.406*
	Sig. (2-tailed)		.049
	N	24	24
A320 Lifetime	Pearson Correlation	.406*	1
	Sig. (2-tailed)	.049	
	N	24	24

*. Correlation is significant at the 0.05 level (2-tailed).

For the relation between knowledge change and aircraft lifecycle, the aggregate number per year of 'maintain', 'update' and 'retire' actions have been plotted. Following bivariate correlation analysis of this aggregate number as dependent variable against the independent variable time (in years), the results can be seen in Figure 6.5 and Table 6.2. There is a significant relation ($p < .05$) of small size (R = .497, R^2 = .247). However, variation in knowledge change per year is notable – knowledge does not change at a constant rate. Again, the effect size is heavily influenced by the variation over the years, especially the dip between 2007-2010.

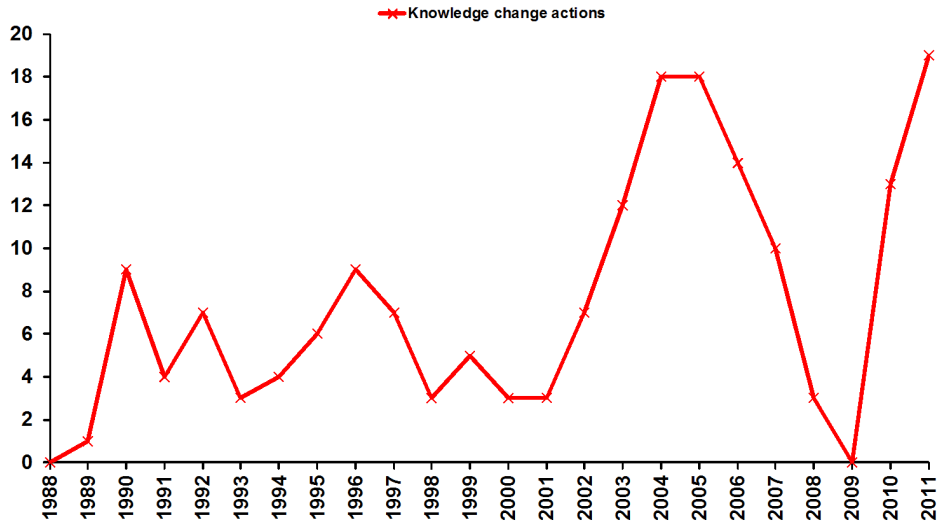

Figure 6.5: Knowledge change versus lifetime (A320)

Table 6.2: Bivariate correlation - knowledge change versus lifetime (A320)

Correlations

		A320 Lifetime	Aggregate knowledge change
A320 Lifetime	Pearson Correlation	1	.524**
	Sig. (2-tailed)		.009
	N	24	24
Aggregate knowledge change	Pearson Correlation	.524**	1
	Sig. (2-tailed)	.009	
	N	24	24

**. Correlation is significant at the 0.01 level (2-tailed).

A final set of tests has been performed to check the frequency of individual knowledge actions (create, maintain, update) per year throughout the aircraft lifecycle. Similar to the preceding analyses, the frequency of actions per year has been enumerated and bivariate regression has been performed for independent variable 'time' and dependent variable 'number of create actions', 'number of maintain actions', and 'number of update actions'. The graphs for the individual knowledge actions are shown in Figure 6.4. For the hypothesized 'time – create' relation, correlation yields insignificant results ($p = .485$), similar to the 'time – maintain' relation ($p = .643$). Only the 'time – update' relation shows a strong significance ($p < .001$) with a medium effect size ($R = .683$, $R^2 = .466$). The 'time-retire' relation has not been tested, given the small set of 'Retire' instances.

B737 Analysis

To analyse the knowledge actions along the knowledge lifecycle, the sample of 648 B737 AD publications from the period of February 1968 until March 2012 has been analysed using a simple frequency count for the number of knowledge actions per category. The results are shown in Figure 6.6. Of the 648 publications, 493 did not move beyond the 'create' knowledge action, whereas 57 publications were maintained, 95 were updated and 3 were retired. Similar to the A320 sample, the number of knowledge actions decreases along the knowledge lifecycle, but a sizeable part of the B737 ADs are subject to knowledge change.

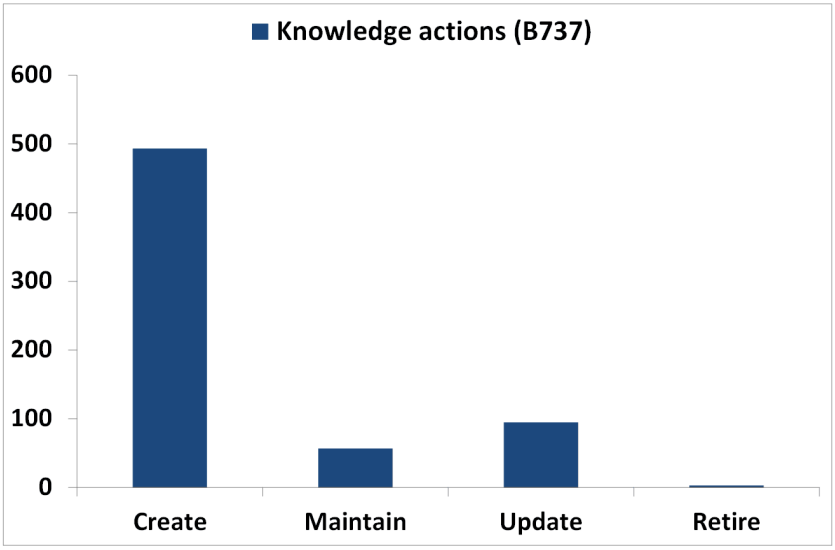

Figure 6.6: Frequency of knowledge actions (B737)

To analyse the yearly number of knowledge actions along the aircraft lifecycle, the combined number of knowledge actions per year for the sample period 1968 – 2012 has been enumerated. The results are shown in Figure 6.7. This data has been subjected to linear bivariate correlation analyses, with either the total number of knowledge actions or the number of individual knowledge actions (create, maintain, update) as dependent variables and time (in years) as the independent variable. For the total number of knowledge actions over time, the results are given in Table 6.3. From Table 6.3, it can be seen that there is a significant ($p < .001$) and medium-strength relation ($R = .765$, $R^2 = .585$) between the total number of knowledge actions and the product lifecycle.

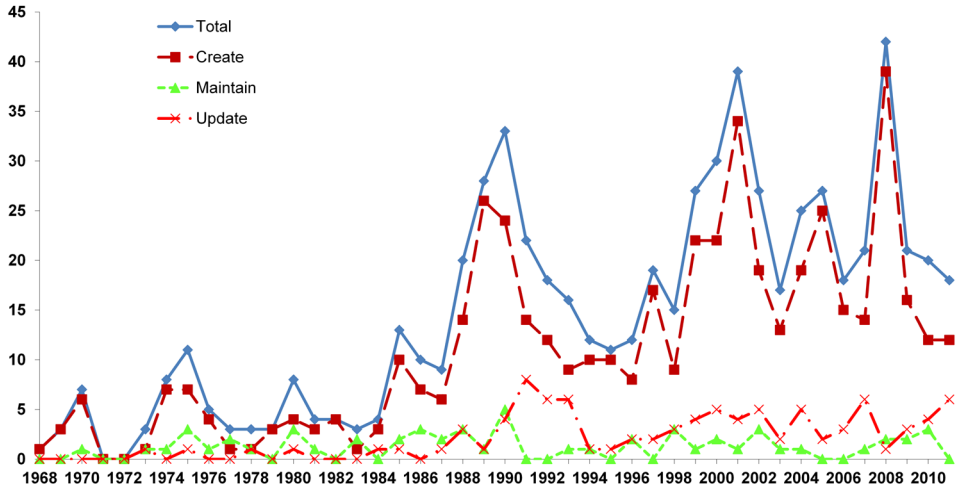

Figure 6.7: Knowledge actions per year versus the B737 lifetime (years)

Table 6.3: Bivariate correlation - knowledge actions per year versus lifetime (B737)

Correlations

		Number of knowledge actions per year	B737 Lifetime
Number of knowledge actions per year	Pearson Correlation	1	.765**
	Sig. (2-tailed)		.000
	N	44	44
B737 Lifetime	Pearson Correlation	.765**	1
	Sig. (2-tailed)	.000	
	N	44	44

**. Correlation is significant at the 0.01 level (2-tailed).

For the relation between knowledge change and aircraft lifecycle, the total aggregate number of 'maintain', 'update' and 'retire' actions have been plotted. Following bivariate regression analysis of this aggregate number as dependent variable against the independent variable time (in years), the results can be seen in Figure 6.8 and Table 6.4. There is a significant relation ($p < .01$) of medium size ($R = .644$, $R^2 = .414$). Similar to the A320 sample, the variation in knowledge change per year indicates a non-constant rate of knowledge change, though the overall trend is an increasing rate of change. The variation has a reducing effect on the effect size.

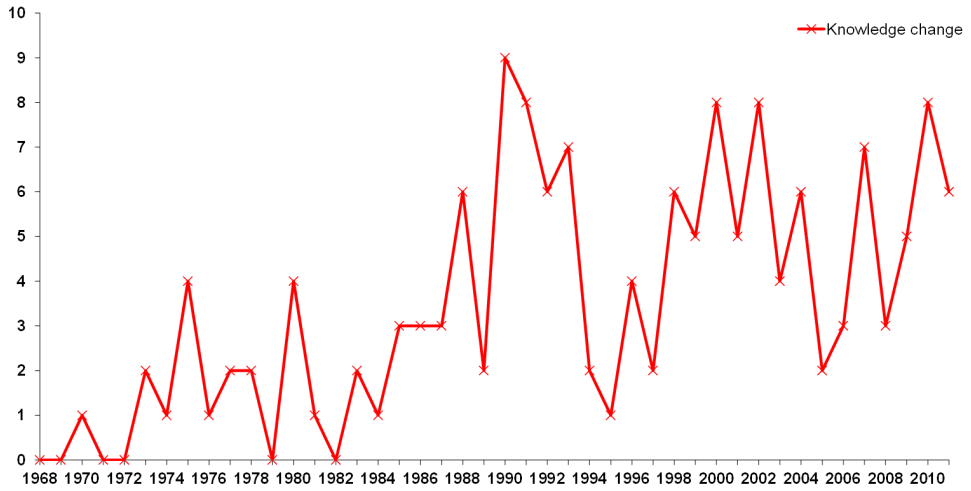

Figure 6.8: Knowledge change versus the B737 lifetime (years)

Table 6.4: Bivariate correlation for knowledge change versus lifetime (B737)

Correlations

		B737 Lifetime	Aggregate knowledge change
B737 Lifetime	Pearson Correlation	1	.644**
	Sig. (2-tailed)		.000
	N	44	44
Aggregate knowledge change	Pearson Correlation	.644**	1
	Sig. (2-tailed)	.000	
	N	44	44

**. Correlation is significant at the 0.01 level (2-tailed).

A final set of tests has been performed to check the frequency of individual knowledge actions (create, maintain, update) per year throughout the aircraft lifecycle. Similar to the preceding analyses, the frequency of actions per year has been enumerated and bivariate regression has been performed for independent variable 'time' and dependent variable 'number of create actions', 'number of maintain actions', and 'number of update actions'. The graphs for the individual knowledge actions are shown in Figure 6.7. Unsurprisingly – given the similar shapes of the 'Create' and 'Total number of knowledge actions' graphs in Figure 6.7 - the regression analysis for the hypothesized 'time – create' relation (Table 6.5) gives significant results ($p < .001$) with medium effect size ($R = .711$, $R^2 = 0.505$). The 'time – maintain' relation (Table 6.6) is not significant at the .05 level ($p = .399$). The 'time – update' relation (Table 6.7) shows a strong significance ($p <$

.001) with a medium effect size (R = .670, R^2 = .449). The 'time-retire' relation was not tested, given the small set of 'Retire' instances.

Table 6.5: Correlation results for knowledge action 'create' versus lifetime (B737)

Correlations

		B737 Lifetime	Knowledge action 'Create'
B737 Lifetime	Pearson Correlation	1	.711**
	Sig. (2-tailed)		.000
	N	44	44
Knowledge action 'Create'	Pearson Correlation	.711**	1
	Sig. (2-tailed)	.000	
	N	44	44

**. Correlation is significant at the 0.01 level (2-tailed).

Table 6.6: Correlation results for knowledge action 'maintain' versus lifetime (B737)

Correlations

		B737 Lifetime	Knowledge action 'Maintain'
B737 Lifetime	Pearson Correlation	1	.130
	Sig. (2-tailed)		.399
	N	44	44
Knowledge action 'Maintain'	Pearson Correlation	.130	1
	Sig. (2-tailed)	.399	
	N	44	44

Table 6.7: Correlation results for knowledge action 'update' versus lifetime (B737)

Correlations

		B737 Lifetime	Knowledge action 'Update'
B737 Lifetime	Pearson Correlation	1	.670**
	Sig. (2-tailed)		.000
	N	44	44
Knowledge action 'Update'	Pearson Correlation	.670**	1
	Sig. (2-tailed)	.000	
	N	44	44

**. Correlation is significant at the 0.01 level (2-tailed).

Consequences of Knowledge Change through Life using A320 and B737 AD Samples

The preceding analysis of the A320 and B737 samples suggests the occurrence of knowledge change in the maintenance domain – airworthiness directives are measurably changing during the lifetime of the respective aircraft. What, then, are the consequences of knowledge change?

From a aircraft through-life perspective, each AD that is in effect already constitutes a knowledge change with respect to the original product (operational) instructions. As such, all individual ADs that are in effect (i.e., the ones that have not been revised, superseded or cancelled) have been analysed for both samples. Does knowledge change, embodied in a created, maintained or updated AD, engender changes in the maintenance domain only? Or does AD knowledge change have an impact beyond the maintenance domain?

To answer these questions, both samples have been analysed for the consequences of through-life knowledge change. Each AD has been checked to see whether it has an impact on:

- **Maintenance:** an AD has an effect on the maintenance domain when inspection and/or repair instructions are mandated. When only inspection requirements are mandated, the AD is simply confined to the maintenance domain. When repair instructions with respect to the affected aircraft parts or systems do not mention the incorporation of new, improved, redesigned, or remanufactured parts, assemblies or systems, the effects of the AD are also confined to the maintenance domain.
- **Flight operations:** a number of ADs mention the update of the Aircraft Flight Manual (AFM) with revised operator instructions. The through-life effect of these ADs is primarily associated with the (flight) operations domain.
- **Manufacturing:** for a few ADs, an unsafe condition was caused by deficiencies in the production processes used for the production of parts and/or assemblies. These ADs mandate changes in the production processes; they consequently have a through-life effect on the manufacturing domain. No part or assembly redesign was incorporated into these ADs.
- **Design:** a few ADs mandate updating the software in flight computers. The associated redesign of flight control software is associated with a through-life effect on the design domain. The manufacturing domain is not involved, as the software can be updated in existing hardware.

- **Design & manufacturing:** a significant number of ADs mandates maintenance on an aircraft using new, improved, redesigned and remanufactured parts, assemblies or systems. For these ADs, the effects stretch through the design and manufacturing domains.

The analysis has been conservative: when an individual AD contains ambiguity regarding the effects on design, manufacturing and/or flight operations, the guideline has been to count the AD as having an effect on maintenance only.

The results of this analysis effort are shown in Figure 6.9 for the A320 sample, and Figure 6.10 for the B737 sample. The majority of knowledge change in maintenance has an effect that is contained to maintenance itself, but a significant minority of knowledge change is associated with through-life consequences.

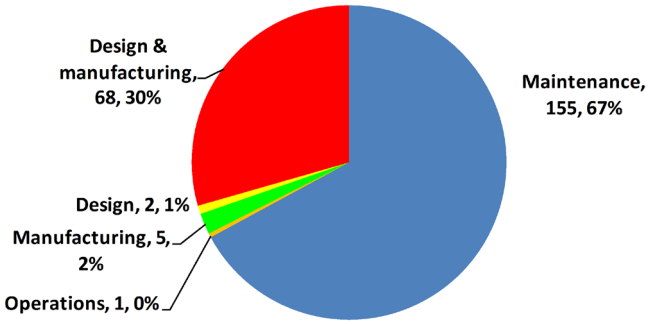

Figure 6.9: Through-life implications of knowledge change (A320 sample)

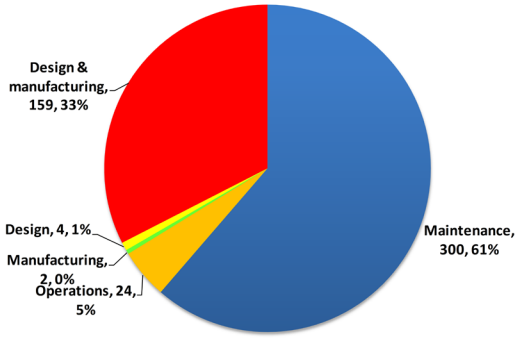

Figure 6.10: Through-life implications of knowledge change (B737 sample)

6.2.2.5 Discussion

The following hypotheses have been posited. The first hypothesis is concerned with the behaviour of knowledge throughout its life:

H_1: The frequency of knowledge actions decreases along the knowledge lifecycle

The remaining set of hypotheses is related to the behaviour of knowledge over the aircraft operational life.

H_2: Number of knowledge actions per year increases during the aircraft operational life

H_3: Use of the 'create' action per year increases during the aircraft operational life

H_4: Knowledge change per year increases during the aircraft operational life

H_5: Use of the 'maintain' action per year increases during the aircraft operational life

H_6: Use of the 'update' action per year increases during the aircraft operational life

The behaviour of ADs during aircraft life can be characterised by using the concept of knowledge actions from the Knowledge Lifecycle Model; all four considered knowledge actions (create, maintain, update, retire) can be identified and measured. With respect to the case-specific hypothesis within the context of the Knowledge Lifecycle Model, the following conclusions can be drawn:

- **Hypothesis 1:** The frequency of knowledge actions does decrease along the knowledge lifecycle.
- **Hypothesis 2:** The number of knowledge actions significantly increases during the aircraft lifecycle, both for A320 and B737, though the effect size for the latter aircraft sample is larger (R_{A320} = .406 versus R_{B767} = .765).
- **Hypothesis 3:** Use of the knowledge action 'create' significantly increases during the aircraft lifecycle for the B737 sample (p < .001, R = .711, R^2 = 0.505). The A320 sample results are not significant.
- **Hypothesis 4:** A significant, small-to-medium size increasing trend in knowledge change per year can be observed from the A320 and B737 samples (R_{A320} = .497, R_{B767} = .644). The underlying rate of knowledge change is not constant.
- **Hypothesis 5:** Use of the knowledge action 'maintain' shows no significant relation with the aircraft lifecycle for both samples.
- **Hypothesis 6:** Use of the knowledge action 'update' shows an increase during the aircraft lifecycle for the A320 sample (p < .001 with a medium effect size (R = .683, R^2 = .466)) and for the B737 sample (p < .001 with a medium effect size (R = .670, R^2 = .449)).

The implications of knowledge change through aircraft life have been quantified. For about 30 to 35% of the airworthiness directives, a through-life implication has been associated.

6.2.3 Application of KLC Ontology: Task Analysis

The modification and inspection task for the B737 slat track main downstop is given in Figure 6.14 as an A-0 IDEF0 diagram. This figure shows the inputs in the form of the Service Bulletin and Airworthiness Directive, the output – a modified and inspected B737 slat track main downstop assembly – and the controls (Airworthiness Directive) and mechanisms (mechanic, tooling). The AD serves as both input and control to the task: it offers input information such as aircraft type applicability and controls the task, for instance through the mandatory compliance time.

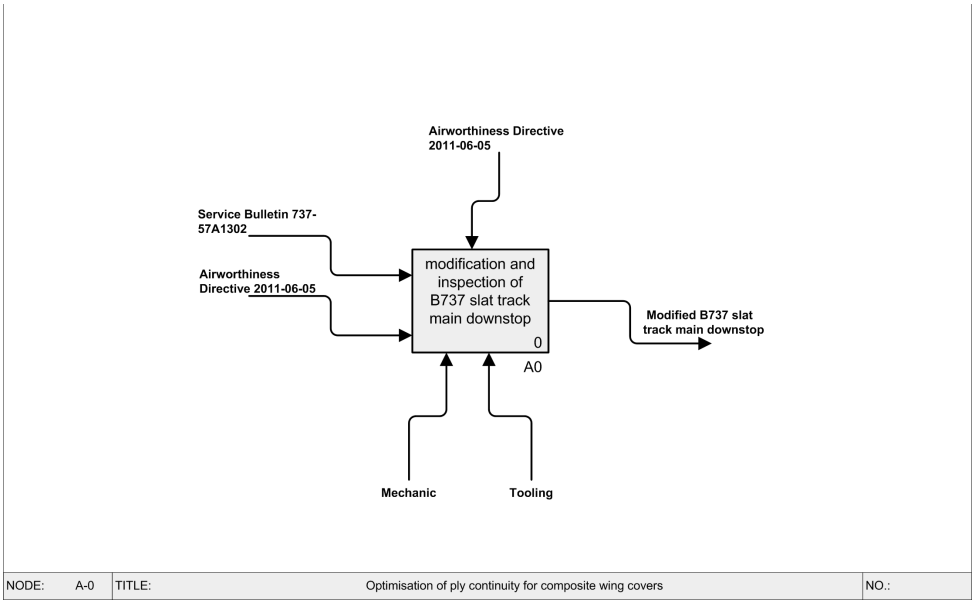

Figure 6.11: IDEF0 A-0 diagram for B737 slat track main downstop modification and inspection task

The subtasks are represented in Figure 6.15 as an A0 IDEF0 diagram, based upon the task description as included in Boeing Service Bulletin 737-57A1302. The first subtask is to obtain access to the assembly, with the service bulletin and wing assembly as inputs and access to the assembly as an output. The task is performed by a mechanic using specific tooling. The second step is to perform preventive modification using the original slat track main downstop assembly and the Service Bulletin, while complying with the requirements of the associated

Airworthiness Directive. After modification, an inspection is performed which results in a maintenance report, which can be used to prove compliance with the AD. The final step is to close access to the assembly and restore the wing to its proper operating conditions.

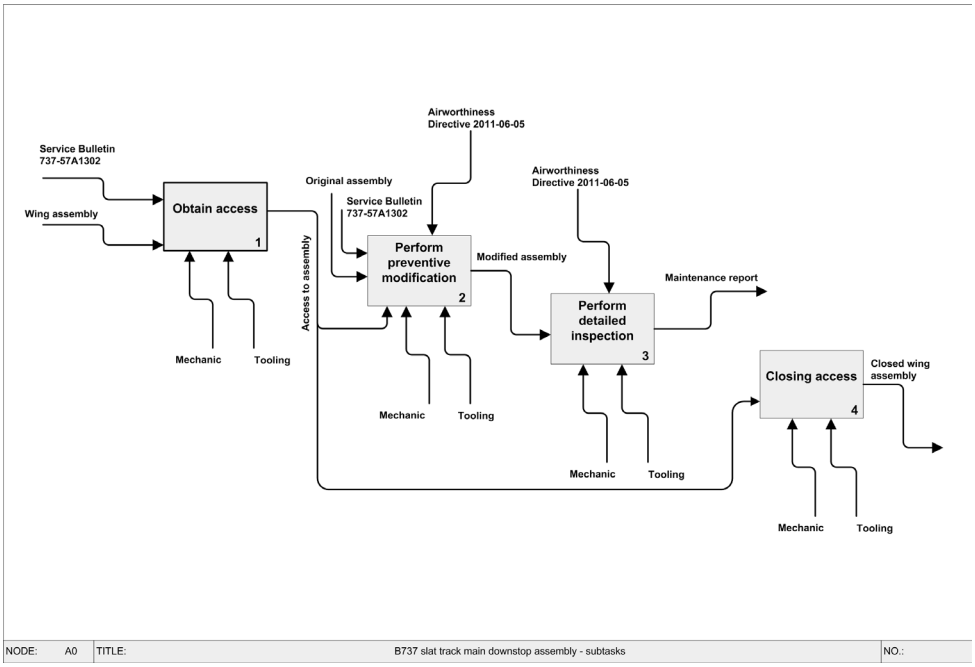

Figure 6.12: IDEF0 A0 diagram for B737 slat track main downstop subtasks

A more detailed description of these subtasks is given as part of the process knowledge description in Section 6.3.1.

6.2.4 Application of KNOMAD: Solution Approach

As for the previous case studies, the KNOMAD methodology as discussed in Section 3.3 is adopted. The KNOMAD steps relative to this case study are shown in Figure 6.13. This figure shows the main KNOMAD steps (Knowledge Capture & Identification of Knowledge Change; Normalisation; Organisation; Modelling & Implementation; Analysis & Delivery) with the associated activities that are required for this particular case study.

The activities are largely similar to those performed in the previous case studies. In the first step (Knowledge Capture & Identification of Knowledge Change), the justification for and scope of the knowledge-based system is established, followed by capture of the knowledge and process elements. Given that knowledge change has been analysed at length in Sections 6.2.1 and 6.2.2, the associated KNOMAD activity can be considered to already having been

applied. For the second step (Normalisation), the focus is on checking data quality and establishing input and output formats. The third step (Organisation) considers development of a domain ontology that holds the relevant concepts and relationships for this particular case study. It is split up into three parts: generation of product, process and resource class diagrams. The fourth step (Modelling & Implementation) concerns the development of models (in the Modelling sub-step), architecture and solution (in the Implementation step). As before, the developed task and domain ontologies are implemented in AKM to support the developed solution. Finally, the Analysis and Delivery steps are combined into one: performance of the solution is assessed relative to the requirements, and the costs and benefits of the solution are explored.

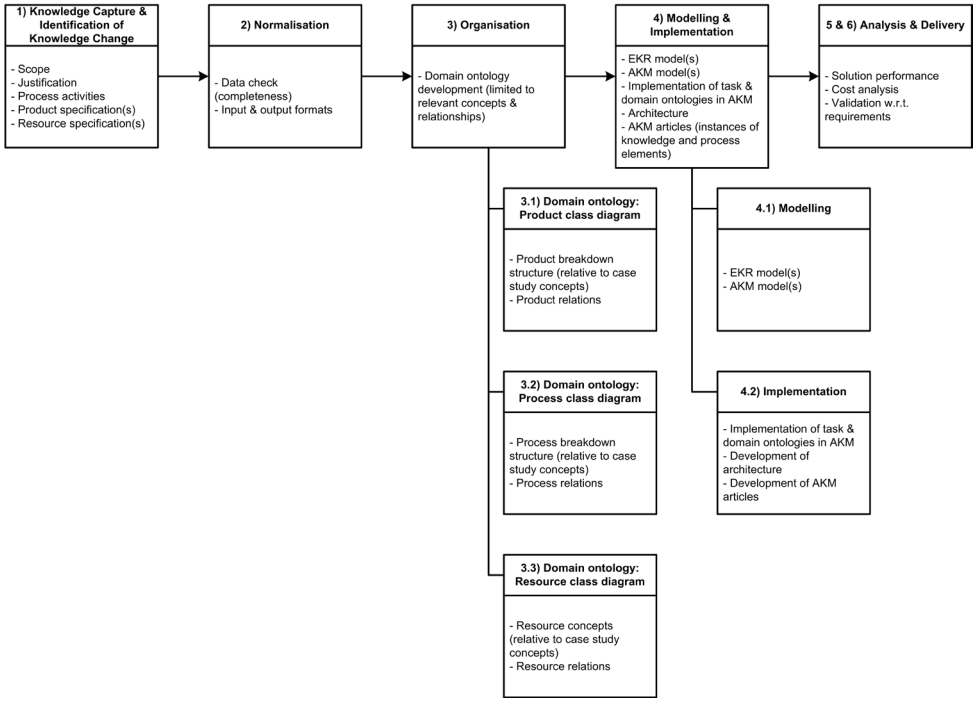

Figure 6.13: Application of KNOMAD to maintenance case study – flow chart

6.3 Results

The next section describes the development of a knowledge-based application for the use case maintenance task: the modification and detailed inspection of the main track downstop of the leading edge slats of most Boeing B737 types. The proof-of-concept solution can cope with knowledge change and addresses issues related to knowledge usability and maintainability.

6.3.1 Knowledge Identification & Capture

The first step in the development of the knowledge-based solution is to capture the required knowledge elements. This knowledge is contained within the relevant ADs and SBs, where the SBs hold the most detailed representation. The knowledge can be captured by considering the product, process and resource dimensions.

- **Product knowledge:** the product considered for this research problem is the main track downstop assembly of the leading edge slats on various Boeing 737 types. The assembly and its main parts are shown in Figure 6.14.

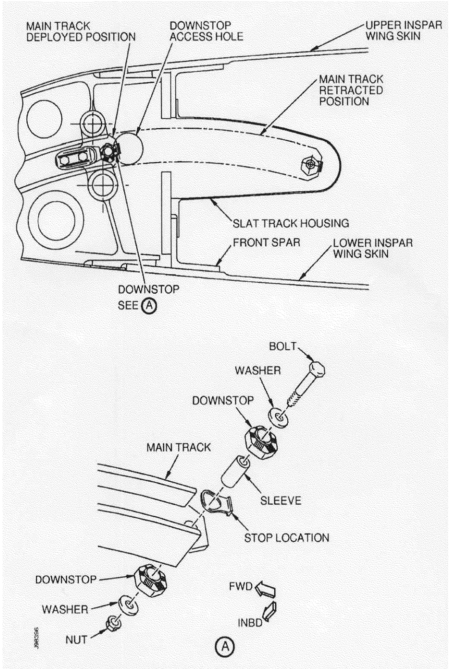

Figure 6.14: Slat main track downstop assembly (FAA, 2011)

When deploying a slat, the slat main track extends to the deployed position by sliding out of the surrounding structure. The main tracks travel through holes in the front spar web when the slat is deployed or retracted. In areas of the wing where fuel is stored, slat track housing (also known as a slat can) is installed on the fuel side of the spar to surround the main track and contain the fuel; this structure protects the fuel tank. Each slat main track has a downstop assembly attached to the aft end of the slat track assembly. The original downstop assembly consisted of a bolt, washer, downstop fitting,

sleeve, stop locator, downstop fitting, washer and nut, as shown in Figure 6.14. The new downstop assembly – as required by AD 2011-06-05 – consists of a new bolt, washer, downstop fitting, another downstop fitting and washer on the opposite end of the hole, and a self-locking nut and pin (both new assembly parts).

In addition to the slat main track downstop assembly, Boeing Service Bulletin 737-57A1302, Revision 1 and AD 2011-06-05 also require the replacement of the original aft side guide bolts with new aft side guide bolts that have drilled heads and are lock-wired together. This is shown in Figure 6.15, where the 'A' indicates the location of the new guide bolts.

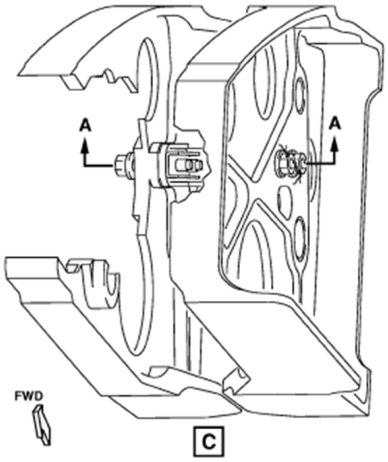

Figure 6.15: Aft side guide bolts (Boeing, 2010)

- **Process knowledge:** the required maintenance process for the modification and inspection of the downstop assembly and aft side guide bolts is described in detail in Boeing Service Bulletin 737-57A1302, Revision 1. It consists of four consecutive steps: obtaining access, carrying out preventive modification and subsequent detailed inspection, and closing access. As may be expected, the second step – preventive modification and subsequent inspection – is the most involved. A simplified overview of the whole process is given in Figure 6.16.

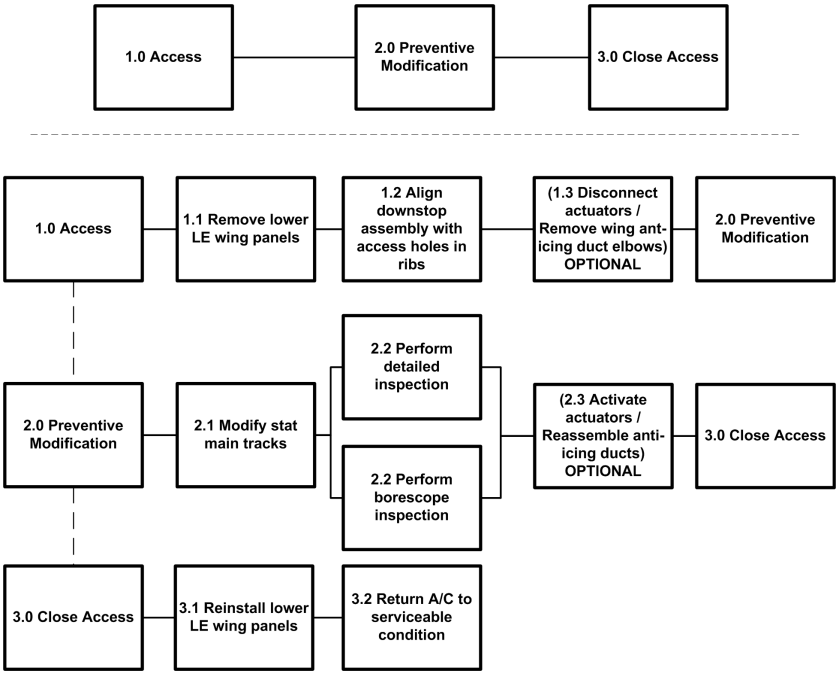

Figure 6.16: Maintenance process for modification and inspection (based on Boeing (2010))

- **Resource knowledge:** the modification and detailed inspection of the downstop assembly and aft side guide bolts requires a varied range of resources.

  First of all, material resources are required, including the new downstop assembly bolts, self-locking nuts and pins, as well as replacements for the downstop fittings and washers. In addition, the new aft side guide bolts and associated lockwires must be used. In total, 32 down stop fittings and washers and 16 bolts, self-locking nuts and pins are required for 16 slat main tracks on a Boeing 737. Additionally, 64 hex head bolts and sufficient lockwire must be used for the aft side guide bolts at 64 locations. Finally, additional materials must be used to carry out the process: sealant, primer, paint, corrosion inhibiting material and corrosion preventive compound.

  In addition to these materials, manpower resources must be reserved for the process. The modification and inspection process requires 18 work hours (FAA, 2007). Obtaining and closing access adds another 17 work hours to the task, for a total of 35 work hours per aircraft (Boeing, 2010).

  The maintenance task does not require special tooling, but optional tooling is offered by Boeing to accomplish the task. Furthermore, when performing a borescope inspection, a borescope is required tooling.

The maintenance task references a number of existing documents, including an Engineering Change Memo, a Boeing Program Letter and the Standard Overhaul Practices Manual (SOPM). It also affects existing documents: the 737 Maintenance Manual (AMM) and Illustrated Parts Catalog (IPC).

6.3.2 Normalization

The knowledge contained within ADs and SBs is subjected to rigorous review and validation. Furthermore, each AD is set up to conform to a specific standard. Given this, no extra effort was necessary to establish traceability and ownership. Furthermore, the accuracy and reliability of the knowledge is assumed to be sufficient.

6.3.3 Organisation

The next step in the development of a solution is to provide a knowledge structure that can be used to store the captured knowledge and can serve as the semantic backbone for the knowledge-based application. Similar to the preceding case studies, a domain-specific set of concepts and relationships has been developed. To elicit the applicable concepts and relationships for the aerospace composite maintenance ontology, various sources have been employed. This includes the regulatory and OEM documents (FAA, 2007; Boeing, 2010; FAA, 2011), as well as research literature (Tsang, 1995; Lampe *et al.*, 2004; Garg and Deshmukh, 2006; Lee *et al.*, 2008; Jagtap and Johnson, 2011; Burhani, 2012).

As before, the high-level concepts of the KLC ontology (**Product**, **Process** and **Resource**) and relationships (see also Section 3.2.3 and Table 3.7) have been used as a starting point for domain ontology development. These concepts of the KLC ontology have been extended into domain-specific class hierarchies. Through the use of the same high-level classes, the resulting maintenance domain ontology shares many concepts and relations with the design and manufacturing domain ontologies (Sections 4.3.3 and 5.3.3). In this section, excerpts of the domain-specific class hierarchies are given to explain how the maintenance domain ontology is composed.

First of all, the domain-specific class hierarchy for the **Product** class is shown in Figure 6.17. The **Assembly** class in this hierarchy has been extended to include the slat assembly (through aggregation), including the slat track assembly and the slat can (track housing) assembly. The former contains the downstop assembly. These assemblies contain parts. To satisfy the requirements of the developed proof-of-concept solution, the parts that make up the downstop assembly (e.g. **Washer**, **Nut**, **Bolt**) have been added to the part hierarchy and the aggregation relationships are shown.

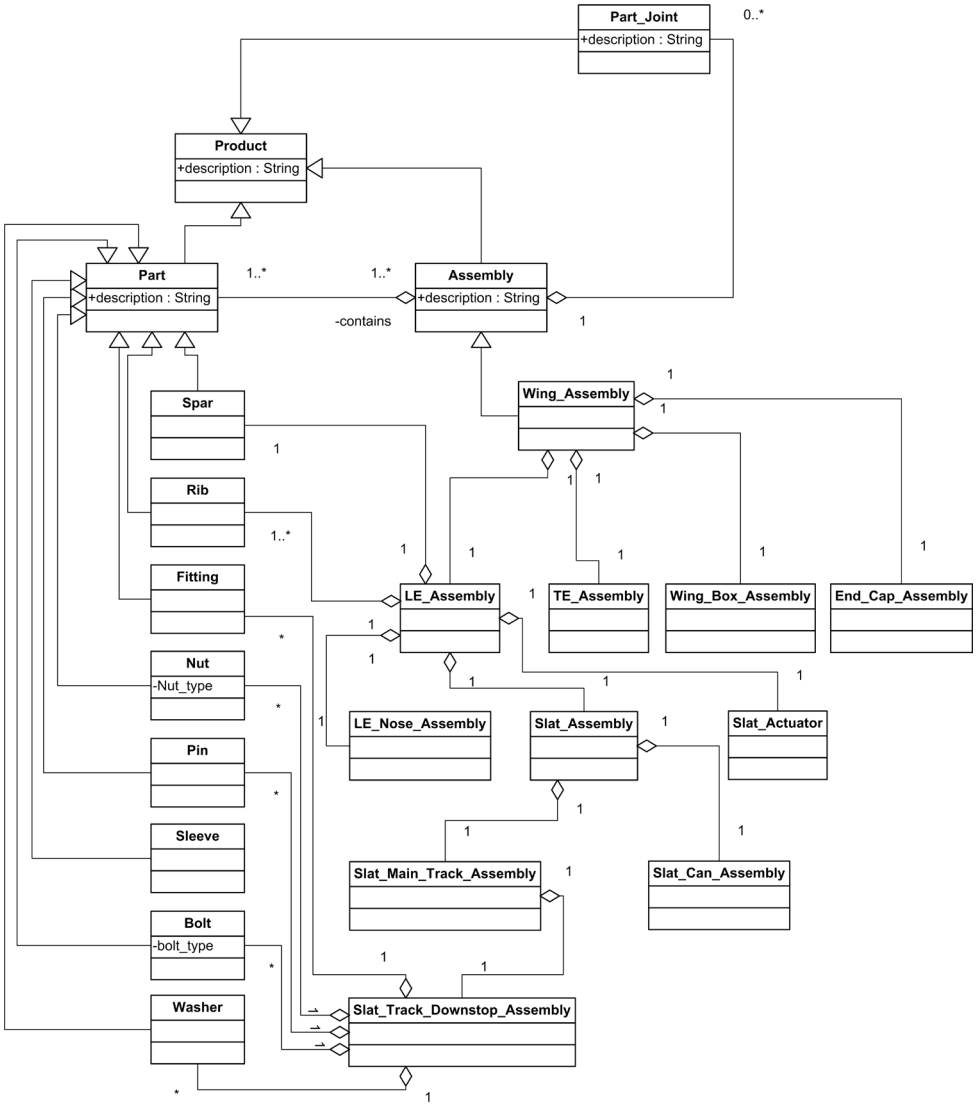

Figure 6.17: extended Product **class hierarchy for the maintenance domain**

Second, the class hierarchy for the **Process** class relative to the maintenance case study is shown in Figure 6.18. The **Process** class has been extended to include **Maintenance_Process**, which in turn is a parent for the **Inspection_Process**, **Modification_Process** and **Repair_Process** classes. As these are all subclasses of the parent **Process** class, they inherit the aggregation with the **Activity** class (in other words, all of the process classes contain one or more activities). The task activities modelled in Section 6.2.3 can be seen as activities belonging to the **Inspection_Process** and **Modification_Process** classes; they have not been added into Figure 6.18.

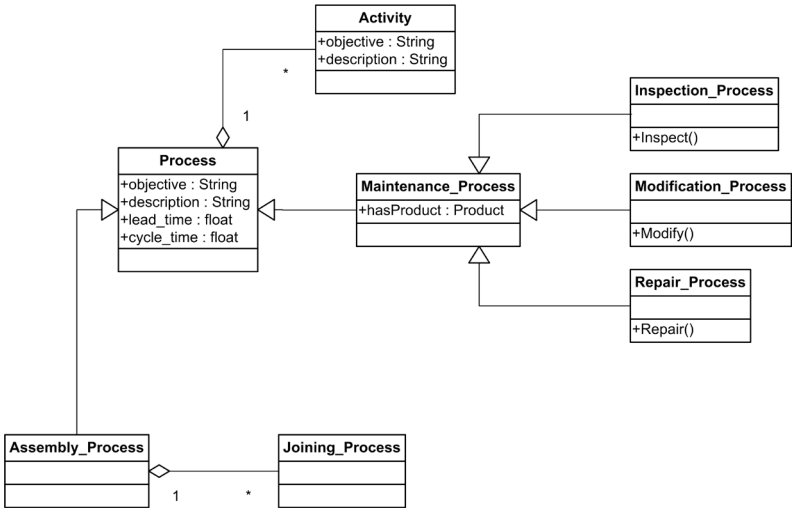

Figure 6.18: extended Process **class hierarchy for the maintenance domain**

The third extension has been made for the **Resource** class (Figure 6.19). The **User_Resource** class has been extended with the **Maintenance_Engineer** and **Mechanic** classes. The **Equipment_Resource** class has been extended with the **Maintenance_Equipment** class. The most significant extension has been made to the **Document_Resource** class. This now includes as subclasses the various document types from the regulator and OEM side.

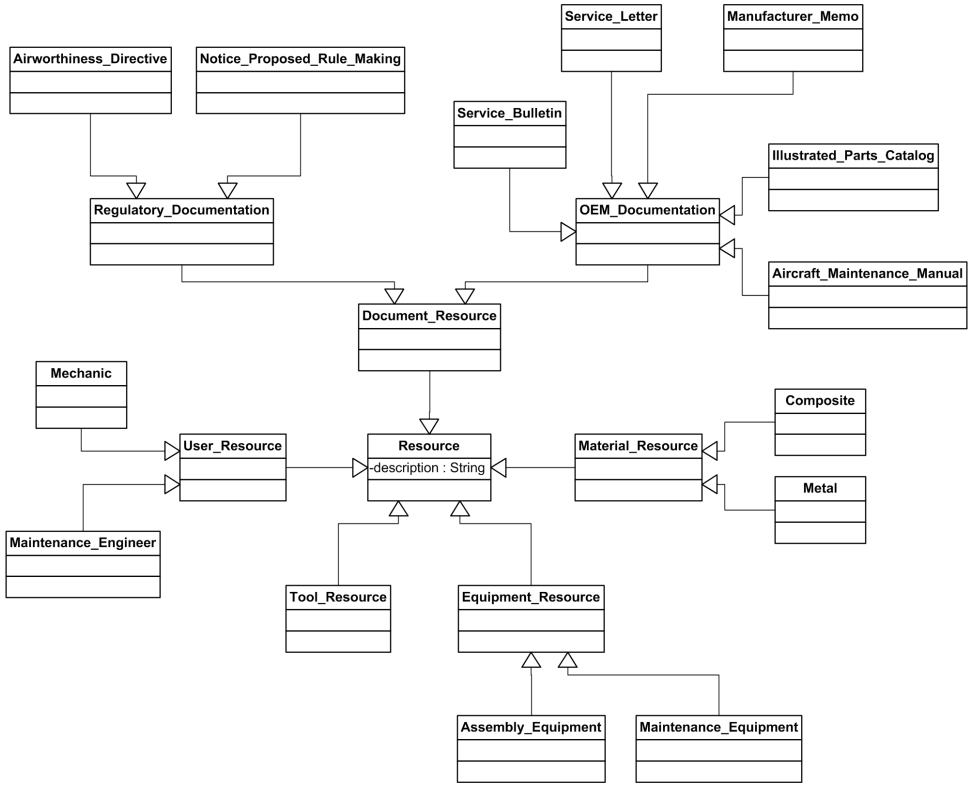

Figure 6.19: extended Resource **class hierarchy for the maintenance domain**

The maintenance domain ontology has been used to structure the captured knowledge and will be used in the subsequent step to annotate (elements of) the developed solution. This is further explained in the following Section.

6.3.4 Modelling & Implementation

This step consists of two activities: modelling of an Enterprise Knowledge Resource (EKR) for the downstop assembly modification and inspection maintenance task, and implementation of the EKR into a functioning solution.

6.3.4.1 Solution Development: EKR Modelling

Similar to the previous case studies, the Enterprise Knowledge Resource concept from the KLC ontology is employed to model and implement a solution. The following EKR classes are considered:

- **Enterprise Knowledge Resource**: the 'container' EKR class retains most of the attributes that are present in the general EKR model (see Figure 3.8). The 'objective' and 'description' attributes have been removed from the

EKR class, as they would be redundant: they have been replaced by including specific maintenance attributes into the class, including effective date, applicability, subject, unsafe condition and compliance time. These attributes have been identified as common attributes in Airworthiness Directives and Service Bulletins.

- **EKR_Knowledge:** the EKR uses knowledge from the related AD(s) and SB(s). The **EKR_Knowledge** class retains the attributes from the general EKR model, and includes the common maintenance attributes (effective date, applicability, subject, unsafe condition, compliance time). The same is true for the **Knowledge_Element** class. Instantiations of the latter class are used to capture knowledge related to the problem, mainly product knowledge such as drawings and specifications.
- **EKR_Process:** the EKR uses a process model to represent the activities that must be completed to comply with the Airworthiness Directive and Service Bulletin. The **EKR_Process** class and **Process_Element** class do not change much with respect to the general EKR model. As in the previous classes, the common maintenance attributes are included into these classes.
- **EKR_Case:** for this use case, a central case repository is set up that can hold the results from the modification and inspection task. Individual case reports are filled into the repository. The class for individual reports has been augmented from its generic representation in the general model (see Figure 3.8) to include maintenance-specific attributes. Besides the common maintenance attributes previously identified, other report-specific attributes such as the maintenance visit number, aircraft registration, flight hours and flight cycles of the aircraft, start date, completion date, task status, order number and order description are included into the class.

Using the preceding considerations, an EKR class diagram has been modelled for this specific case study and associated task. The UML class diagram is shown in Figure 6.20. It incorporates the EKR classes and attributes mentioned above.

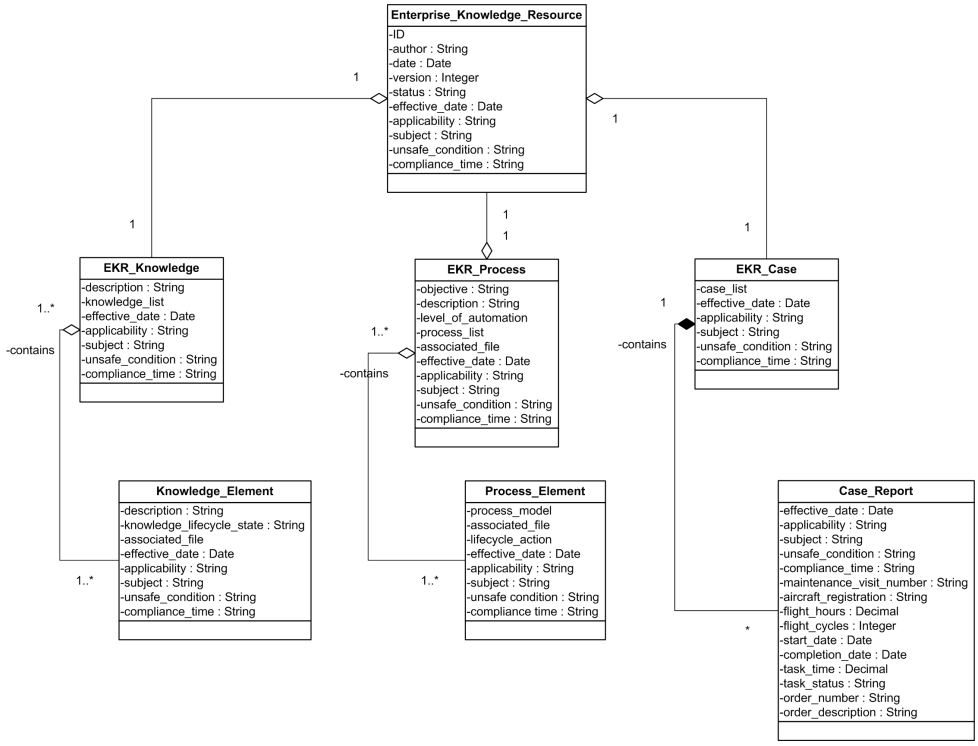

Figure 6.20: EKR class diagram (UML) for maintenance case study

An EKR (and its subsidiary elements, such as knowledge elements) are annotated using the previously introduced domain ontology. The resource, product and process hierarchies together offer the necessary classification richness to annotate an EKR: using these classification hierarchies allows for unique combinations of semantic elements for annotation of a specific EKR, supporting search and retrieval by end users. It will be shown in the next section how this is achieved.

6.3.4.2 Solution Development: EKR Implementation

One EKR has been implemented for this maintenance domain case study: the modification and inspection of the main track downstop assembly. To implement the EKR and associated models presented in the previous section, a solution has been developed on the basis of the Ardans Knowledge Maker (AKM) knowledge management tool.

Based on the developed UML ontologies, a number of AKM models have been developed for the EKR class and its subsidiary classes (knowledge, knowledge element, process, process element, case and case report). For each class, a single model is made that contains fields. These fields represent the attributes of the classes. The relations between the classes are represented

through the addition of direct links between related AKM models. Some automated functionality is added by using the XPATH query language to identify and fill model fields by retrieving node information from the XML data that comes from MySQL. For instance, XPATH expressions are used to let the knowledge, knowledge element, process and process element models inherit the common maintenance attributes (effective date, applicability, subject, unsafe condition, compliance time) from the EKR container class. Furthermore, metadata such as author, date and status is automatically added. XPATH is also used to facilitate the implementation of 'templates' that guarantee a consistent representation of model instances. An example of an AKM model (representing the **Knowledge_Element** class) is given in Figure 6.21. The model in this Figure inherits some metadata automatically (author, date, version, status, effective date, applicability, subject (ATA), unsafe condition, compliance time, and associated EKR; see also Figure 6.23). Other knowledge needs to be filled in manually: description, lifecycle state and associated file(s).

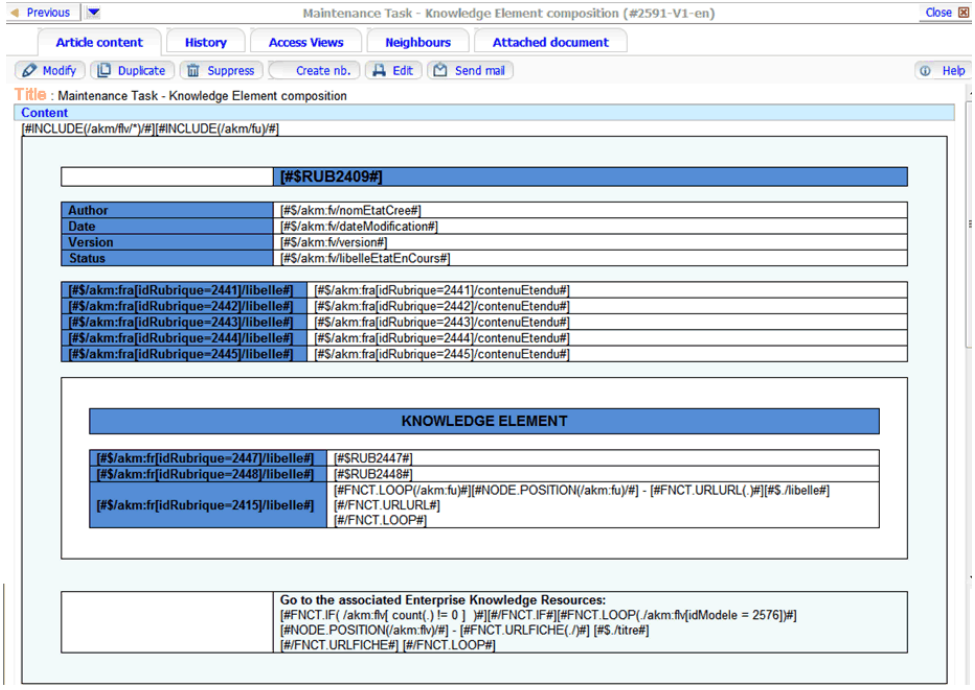

Figure 6.21: AKM model for the Knowledge_Element **class for maintenance case study**

The AKM models are used to generate knowledge articles; they are in effect instances of the EKR classes implemented in AKM. The process of creating articles and generating the article content is currently largely manual. The AKM models take away much work by offering a consistent representation and inheriting article fields related to common maintenance attributes automatically. However, the remaining fields must be filled manually with the appropriate knowledge. The following figures give examples of implemented EKR, knowledge element, process element and case report articles for this case study.

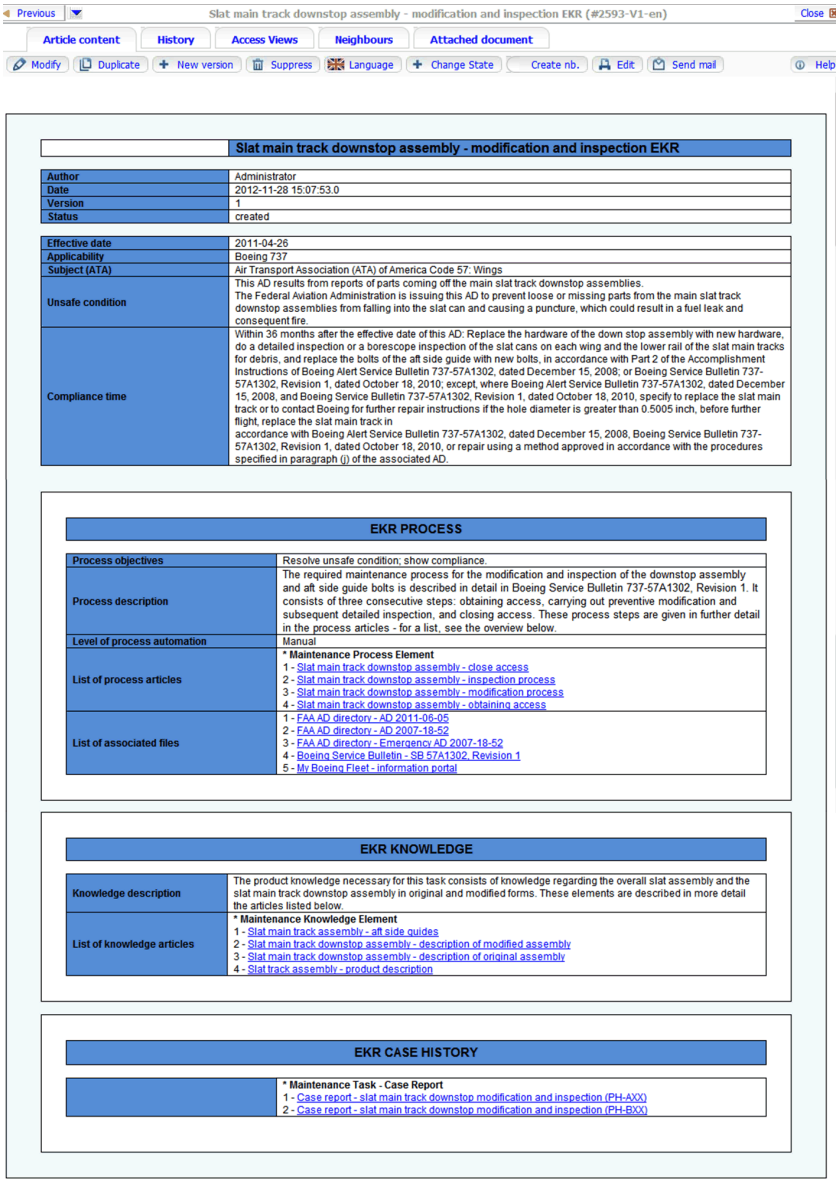

Figure 6.22: Example of EKR article for maintenance case study

Figure 6.22 shows the EKR for the case study engineering task (i.e., the slat track downstop assembly modification and inspection EKR). The metadata (author, date, version, status) is filled in automatically. The common maintenance attributes (effective date, applicability, subject (ATA), unsafe condition, compliance time) are filled in manually. The EKR further consists of the associated process elements (under 'EKR Process'), knowledge elements (under 'EKR

Knowledge') and case reports (under 'EKR Case History'). The process elements have been associated with the relevant documentation, including the ADs and SB.

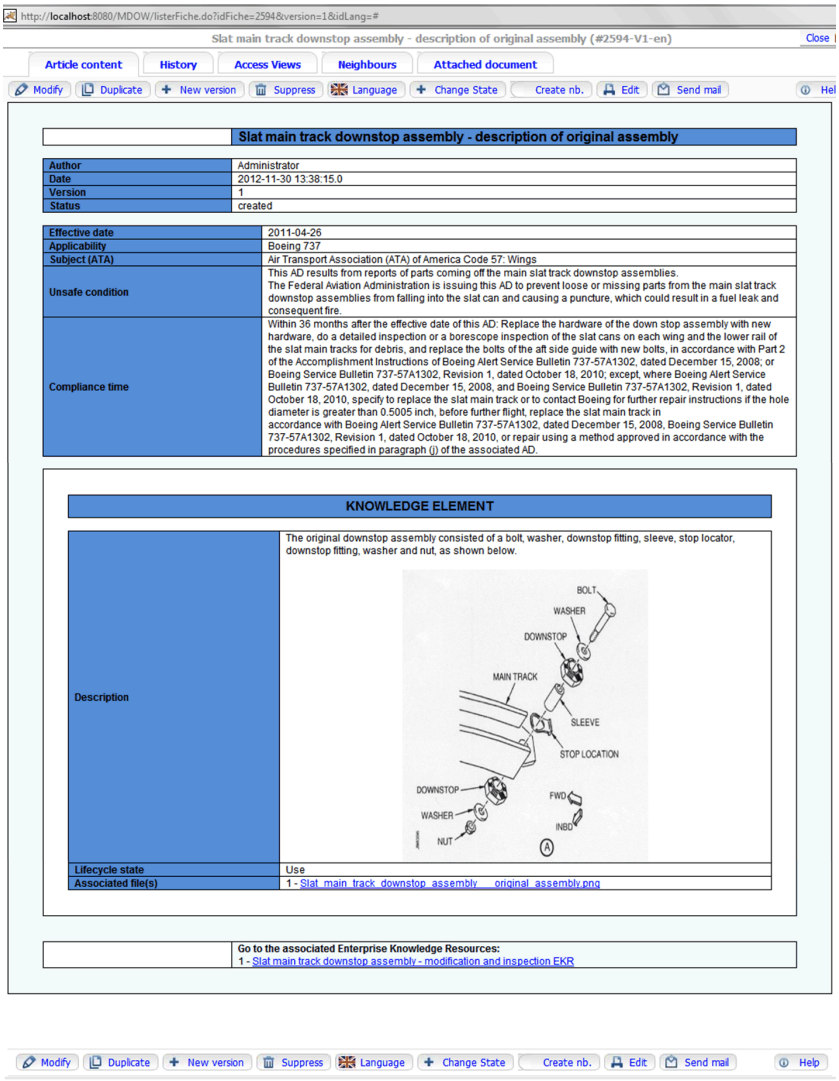

Figure 6.23: Example of knowledge element article

Figure 6.23 shows an example of a knowledge element belonging to the maintenance task EKR: a description of the original slat main track downstop assembly. As previously, the metadata is filled in automatically upon creation. The common maintenance attributes are inherited from the EKR superclass. The description, lifecycle state and associated file(s) of the knowledge element have been filled in manually.

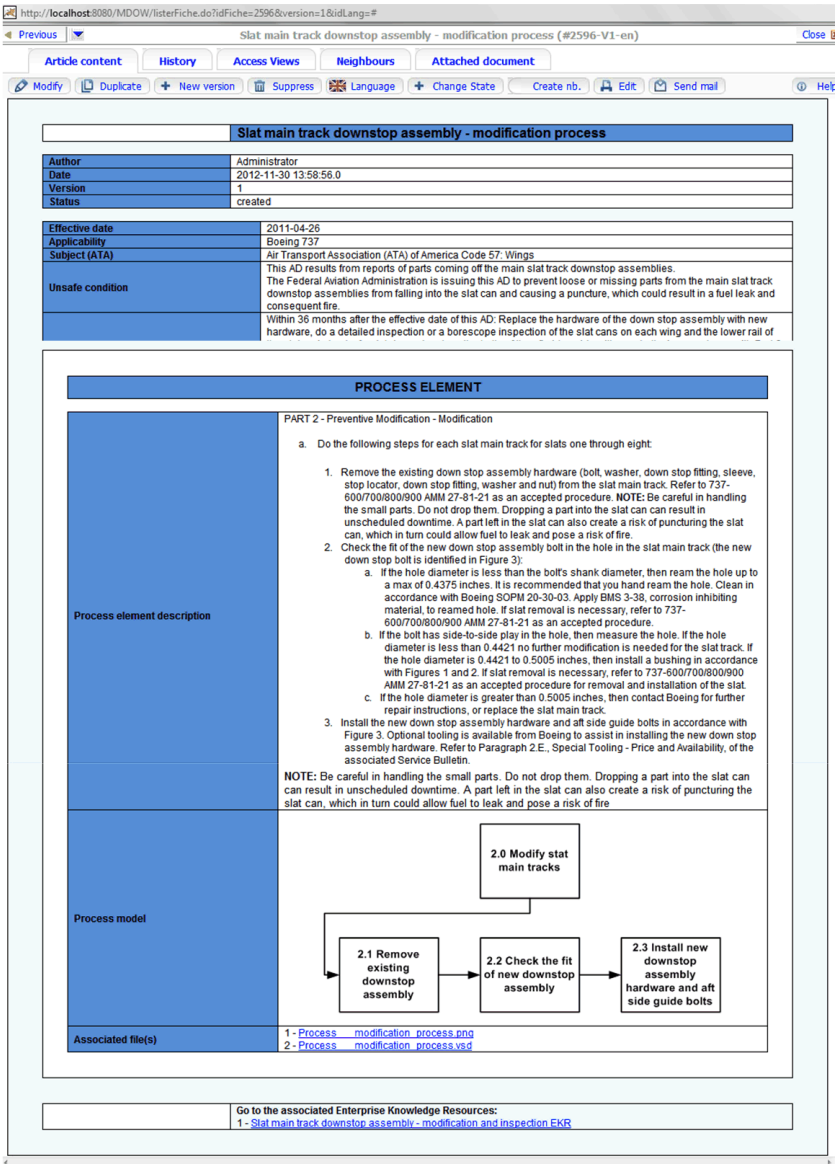

Figure 6.24: Example of process element article

Figure 6.24 shows an example for a process element belonging to the maintenance task EKR. Similar to the knowledge element described in Figure 6.23, the metadata is filled in automatically upon creation of the article and the common maintenance attributes are inherited from the associated EKR. The description, process model and associated file(s) fields have been filled in manually.

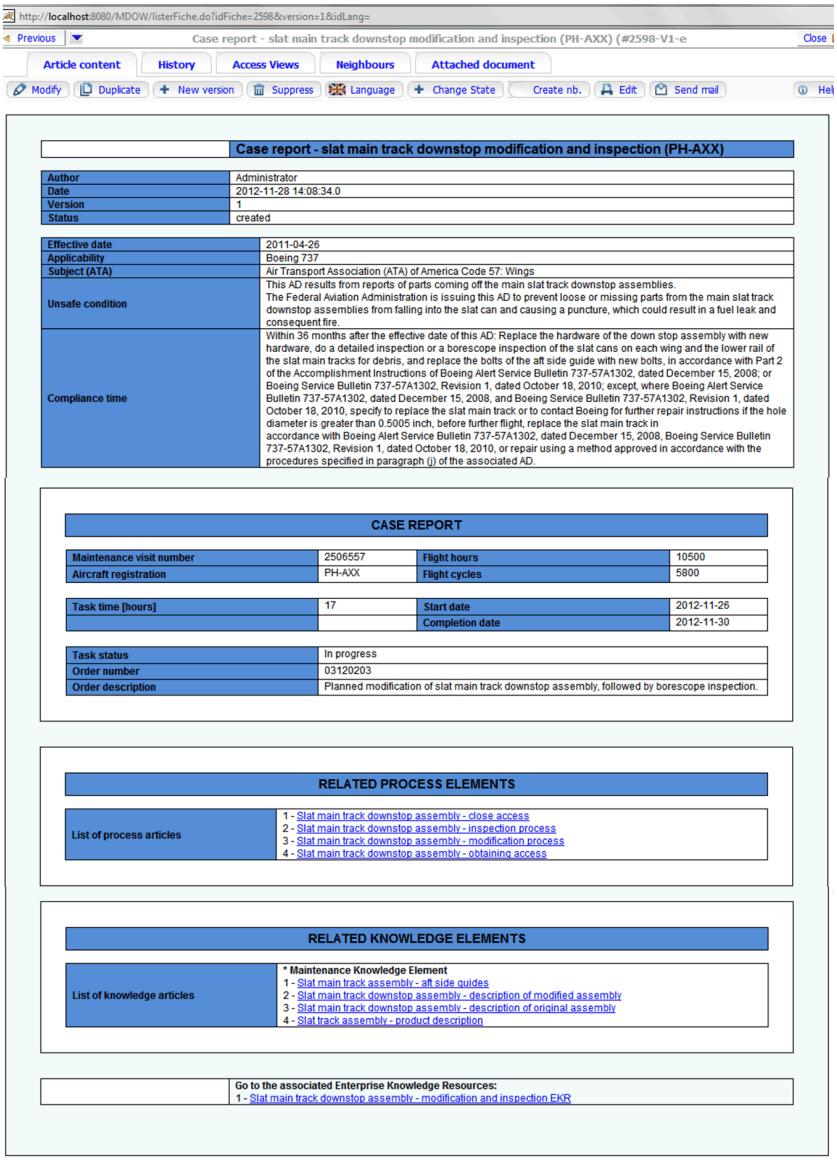

Figure 6.25: Example of case report article

Finally, Figure 6.25 shows an example of a case report article for the maintenance task. Like the knowledge and process element articles, metadata and common maintenance attributes are inherited. Furthermore, associated knowledge and process elements are linked to the case report using the 'Neighbour article' functionality of AKM. The case report furthermore contains

some report-specific attributes (under 'Case Report'), such as maintenance visit number, aircraft registration, task time, flight hours, flight cycles, and others.

The case report model and associated articles are particularly important from the perspective of documentation management for maintenance compliance. The format of these case reports can easily be changed to fit company specifications. The AKM tool includes functionality to export articles and article information into Word or Excel directly. This makes it possible to completely digitalize the generation, storage and management of maintenance documentation.

To enable the search and retrieval of EKRs in the maintenance domain, semantic annotation is used. Annotation of EKRs and its subsidiary elements is achieved through applying the PPR maintenance domain ontology concepts and relationships to the EKR classes. An example for the slat main track downstop assembly EKR is given in Figure 6.26. This Figure shows the Product-Process-Resource classes that have been used to annotate the EKR. Similarly, the knowledge elements, process elements and case reports are annotated by Product-Process-Resource classes, but this is not shown in the Figure.

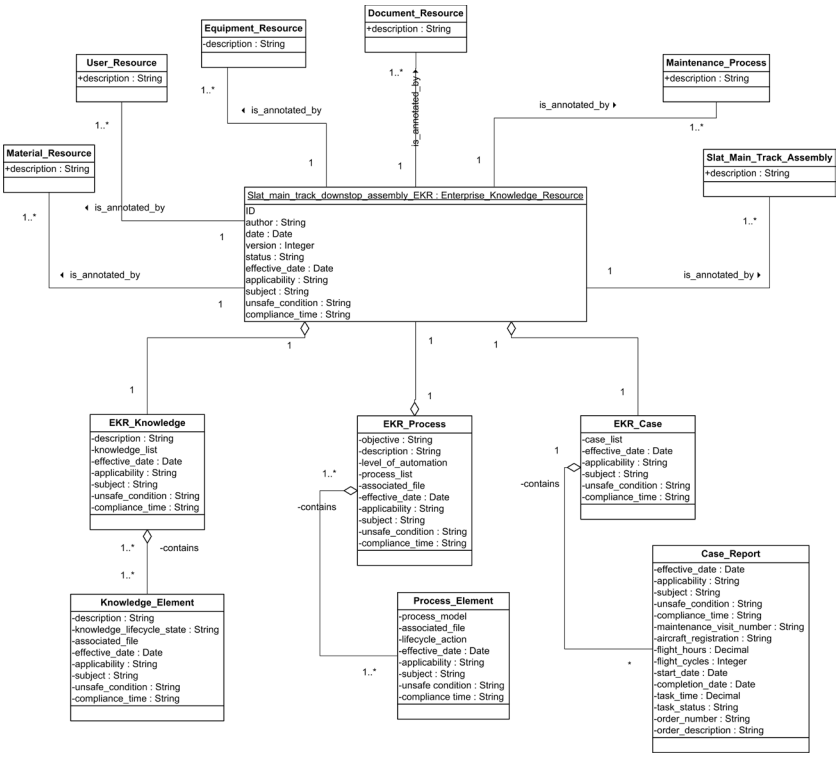

Figure 6.26: Semantic annotation of EKR

In implemented form, annotation is achieved through article tags in AKM, which associate an article (be it an EKR article, a knowledge element article, a process element article or a case report) with a number of semantic tags. An example of tagging the downstop assembly EKR is shown in Figure 6.27, which shows checkmarks for each tag that has been applied (e.g. Product: Slat Main Track Assembly and Document Resource: Service Bulletin).

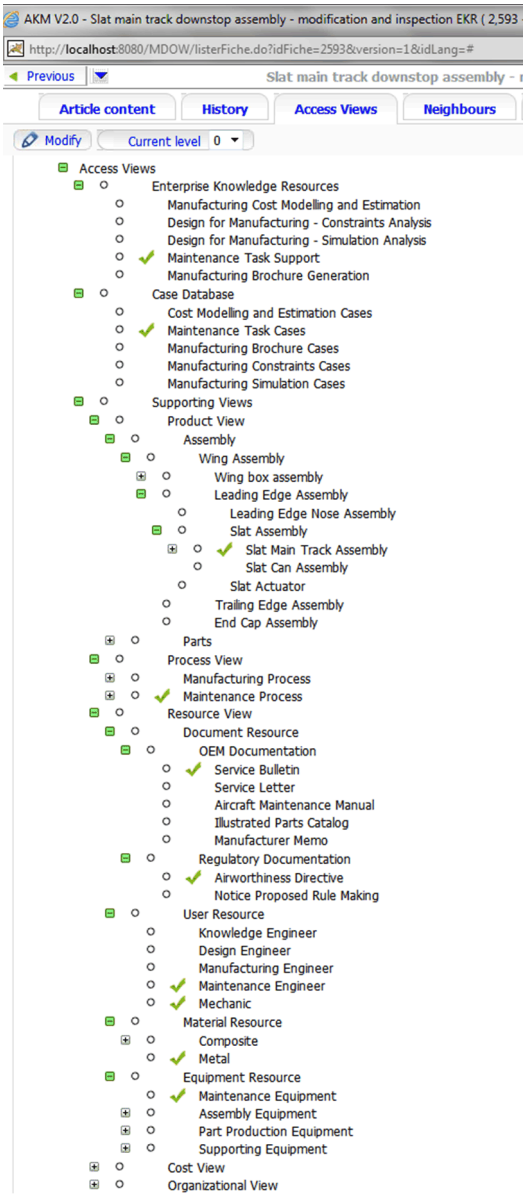

Figure 6.27: Tagging an EKR in Ardans Knowledge Maker

6.3.5 Analysis & Delivery

The Knowledge Lifecycle Model, the KLC ontology and the KNOMAD methodology have been applied to construct a proof-of-concept knowledge-based application to support the execution and reporting of a maintenance task: B737 leading edge slat main track downstop assembly inspection and modification. The solution does not include an inference capability, nor is significant automation of existing processes achieved at this point. However, through the use of the EKR approach, the solution supports capture, formalization, use, maintenance of knowledge – the central aspects of the overall dissertation research objective as well as this case study.

The solution meets the following case study requirements:

- It supports end users in the execution of a maintenance task.
- It provides a digital means of record keeping.

The implemented proof-of-concept solution consists of a single EKR. The implementation of more EKRs has not been considered for this case study, as a) only limited access to OEM Service Bulletins was available, b) the functional solution has been developed to a proof-of-concept stage to meet the case study objectives.

As such, quantification of costs and benefits of the tool has not been considered in detail. Development of EKR models and model XPATH code has taken approximately one working day. Capturing the illustrations and text from the ADs and SBs for the slat main track downstop assembly modification and inspection, followed by the development of EKR articles (EKR, knowledge and process elements), has taken approximately half a working day. It can be estimated that development of a full EKR will require somewhere between half a working day and a full working day. As an indication: 16 ADs have been issued for B737 maintenance tasks related to the aircraft wing.

No delivery aspects are considered, given the lack of collaboration with an industrial partner for this case study.

6.4 Discussion of Results

The combination of the ontology-based approach and use of the AKM tool addresses the usability and maintainability requirements associated with this case study in the following way:

- **Moving beyond black-box applications and ensuring transparency:** the developed EKR brings together dispersed maintenance knowledge (ADs, SBs, maintenance reports) into one access point. The automatically

included metadata as well as the semantic annotation coupled with AKM functionality makes it straightforward to access, inspect and maintain EKR elements. The solution enables storing, justifying and updating maintenance knowledge elements and processes and supports maintenance record keeping. With respect to transparency, the use of the ontology-based EKR approach makes it clear which knowledge is involved for specific maintenance tasks: the knowledge elements, process elements and case reports are all gathered within the EKR 'container' and are captured in a standard way. These elements and reports can be straightforwardly searched and retrieved through the semantic annotation as well as the article metadata which is automatically added upon creation of an article.

- **Task orientation:** knowledge involves a 'capability for effective action'. The capability for action is met in two ways. Firstly the solution offers support for maintenance task execution by offering a 'one-stop' portal for the documentation, process models and maintenance reports associated with that task. Secondly, end users can use the web-based tool to create and manage maintenance reports.
- **Expert / end user involvement:** through the ontology-based EKR approach, end users are able to identify, use, interact with and if necessary, maintain or update the data, information and knowledge related to a specific maintenance task.

There are a number of disadvantages and challenges related to the currently implemented solution. First of all, the solution requires a relatively high amount of manual interaction, primarily in setting up EKRs but also in completing maintenance reports.

Despite the relatively low time needed to implement a single EKR, the sheer amount of ADs and SBs available indicates a large investment of resources to set up a complete knowledge base with EKRs for each maintenance task. There is however some potential to automate knowledge article generation by linking AKM with information retrieved from myboeingfleet.com and FAA / EASA databases. This is because information in XML format can be imported to and exported from AKM. The completion of case reports also requires manual input. Similar to the previous point, case report generation is technically possible by linking AKM with external maintenance programs. However, these options have not explored (yet) as they are beyond the objective and scope of this case study.

Besides additional insight into the application of the ontology-based approach, the maintenance case study has offered further insight into the Knowledge Lifecycle model through quantification of a sample of ADs. The

associated analysis has shown that the Knowledge Lifecycle Model and the concepts of knowledge states and knowledge actions can be used to characterise and quantify the behaviour of knowledge throughout its lifecycle. With respect to the associated analyses, a few essential limitations must be noted:

- The preceding analysis has not considered the measurement of knowledge behaviour throughout aircraft lifecycle stages: the changes in knowledge from design to manufacturing and maintenance have not been considered.
- Though the knowledge actions can successfully be applied to measure knowledge behaviour, the maxim of 'correlation, not causation' still applies; underlying causes of observed behaviour are not analysed and explained here.

With respect to the development of the Knowledge Lifecycle Model itself, a potential development is the further operationalization of knowledge states and actions – it may be possible to establish quantitative indicators for the 'performance' of knowledge throughout its life. Furthermore, the influence of knowledge types must be established. It may very well be the case that some knowledge types are more sensitive to knowledge change than others. Both of these developments are not considered further within the scope of this dissertation.

7 Conclusion

In this final chapter, the findings from the three case studies are synthesized and the overall contribution to theory is discussed. This leads to conclusions with respect to the research objectives and questions, given in Section 7.2. Finally, the limitations of the performed research will be discussed, followed by recommendations for future research.

7.1 Research Synthesis

Three case studies have been performed in the design, manufacturing and maintenance phases of the aircraft lifecycle to validate the theoretical contributions and to reach the objective of supporting consistent formalization, use and maintenance of changing knowledge. The contributions from theory development and the case studies can be synthesized into a vision for knowledge change in knowledge engineering and into a reflection on the results in relation to the research objectives and challenges.

7.1.1 Synthesizing a Vision for Knowledge Engineering

When synthesizing the theoretical contributions and the case study contributions, a vision emerges with regard to knowledge engineering for static (routine) processes versus dynamic processes in which knowledge is subject to change. This vision is encapsulated in Figure 7.1, which highlights two streams of knowledge engineering with associated methodologies and models. Stream 1 concerns static knowledge, i.e. knowledge that does not change or changes slowly and predictably; this knowledge is associated with routine processes, can be captured, mapped and used to automate engineering processes to a very high degree through the use of knowledge-based systems and applications. Stream 2 concerns dynamic knowledge, i.e. knowledge that changes. This knowledge is associated with changing processes, can also be captured, mapped and used for knowledge-based systems and applications, but care must be taken to account for knowledge change: attention to maintainability and usability contribute towards continued use. Figure 7.1 expresses that the Knowledge Lifecycle Model has the potential to (quantitatively) assess knowledge change in engineering processes. Based on the outcome of such an assessment, the methods, models and tools necessary for the development of a knowledge-based system can be chosen. For routine (static) processes, available methodologies such as MOKA and CommonKADS and associated models (MOKA Informal & Formal models; CommonKADS' Knowledge Model) can be adopted to automate engineering work. In Figure 7.1, the MOKA methodology and models are given on the left-hand side in the 'Routine processes' box. The CommonKADS set of models is given on the right-hand side in

that box. For dynamic processes, the method and model proposed in this dissertation (KNOMAD + KLC ontology) can be used – they are given in the 'Dynamic processes' box. KNOMAD and the KLC ontology can be supplemented by models from stream 1 (e.g. the use of CommonKADS' task and inference templates from its Knowledge Model to support development of the KLC ontology's **Process_Element** and **Process** classes). The potential interfaces between methodologies, models and tools for static and dynamic knowledge processes have intentionally been represented with a single dotted line between the static and dynamic process 'regions' under point 2). This dotted line is representative for the realization that issues such as decision variables (when is a knowledge process 'sufficiently' dynamic or static to make a choice for a certain solution approach) and interaction between methodologies and models have not yet been researched in detail.

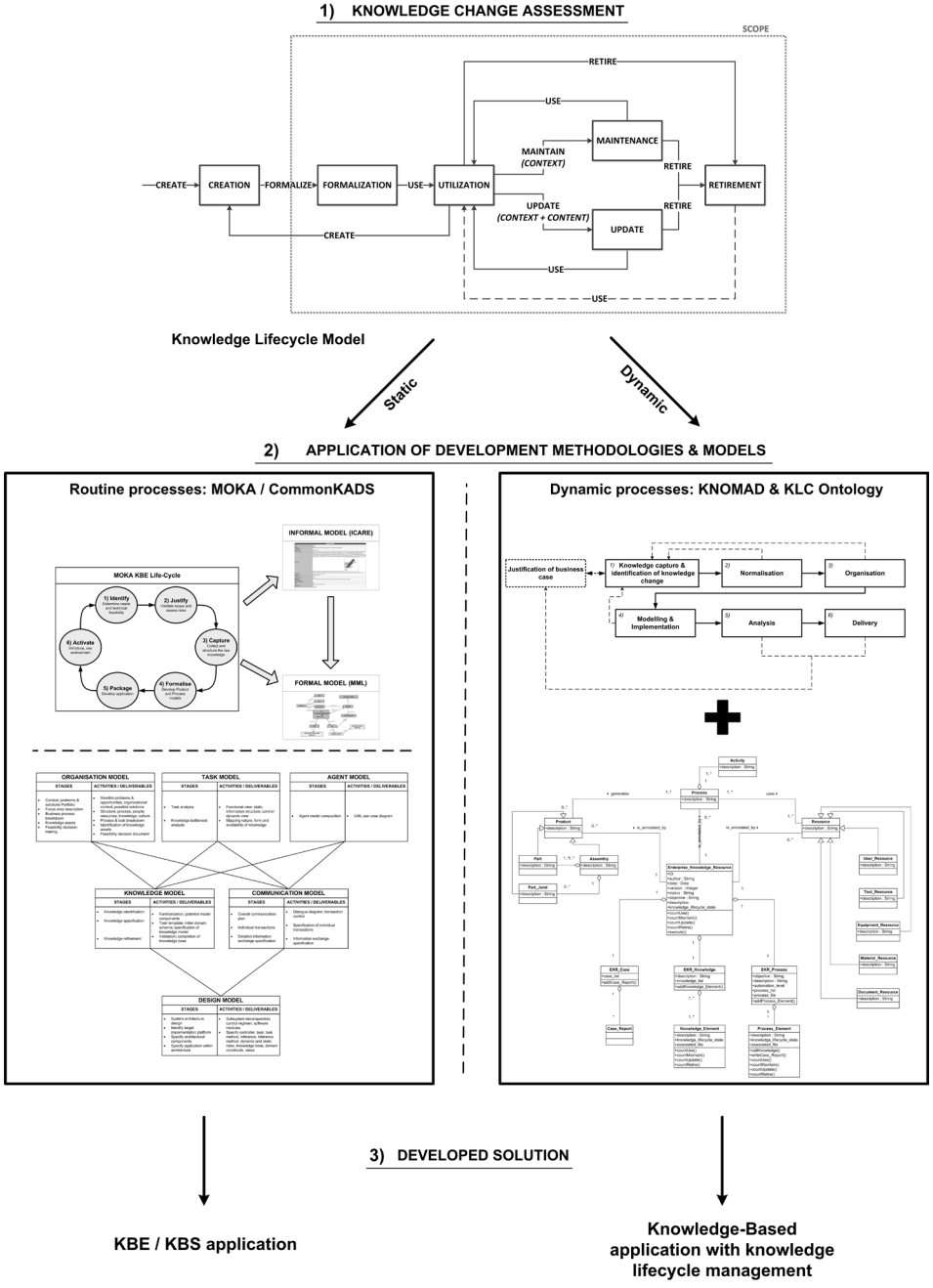

Figure 7.1: Two streams of knowledge engineering related to knowledge change

7.1.2 Synthesizing the Case Study Results relative to Research Objectives and Challenges

Seen as a whole, what insights do the case studies offer with respect to the formalization, use and maintenance of changing knowledge? How does the use of the Knowledge Lifecycle Model, KLC ontology and KNOMAD meet the common research challenges identified in Section 2.3? For convenience, the research objectives and associated challenges are summarized in Table 7.1.

Table 7.1: Research objectives related to research challenges

Research objective	Associated research challenge(s)
Knowledge lifecycle modelling	Characterise, model and quantify the behaviour of knowledge within product life
Ontology-based approach to support knowledge change	Maintainability: - Moving beyond black-box KBS applications and ensuring transparency Usability: - Task orientation - Expert / end user involvement
Methodology development	Methodological approach to facilitate knowledge change management

With respect to the theory-oriented challenge – *characterisation, modelling and quantification of the behaviour of knowledge within aircraft life* – the following observations can be made from the case studies. The Knowledge Lifecycle Model has been developed to characterize and model the knowledge lifecycle by incorporating the concepts of knowledge states and actions. This model has been incorporated in the KNOMAD methodology and has successfully been applied to identify and characterize knowledge within three aircraft lifecycle phases. For the design and manufacturing domains, a qualitative discussion of knowledge change using the KLM concepts has proven possible. For the maintenance domain, both qualification and quantification of knowledge change have been performed.

One challenge regarding maintainability has been identified: *moving beyond black-box KBS applications and ensuring transparency*. In all three case studies, it has proven possible to set up solutions that allow knowledge and processes to be inspected and modified by knowledge engineers and end users. Through the **Knowledge** and **Knowledge Element** classes, users can inspect individual knowledge elements, both in informal (e.g. maintenance task knowledge, see Section 6.3.4.2) and combined informal – formal representation (e.g. manufacturing constraints for ply continuity optimization, see Section 4.3.4.2 and Figure 4.9). The **Process** and **Process_Element** classes offer the user the possibility to inspect the task activities and inferences that are used to solve a problem. For

example, the manufacturing cost model has been split up into process steps that can be individually inspected, downloaded and used to compose and run a cost model.

The cases have shown how a combination of inputs (knowledge), process, and outputs (cases) may be employed. Through the EKR concept, traceability is ensured. In particular, the **Case** class and the associated case reports enable tracing the outputs of knowledge application for a specific task, as well as tracing the knowledge and processes used to perform a task. The metadata that is associated with knowledge and process elements (authorship, lifecycle state, status, etc.) also aids traceability in terms of knowledge ownership, validity and reliability. Through the PPR paradigm, visibility of key concepts is ensured. It has been shown in the three case studies how development of a domain ontology in combination with semantic annotation of implemented EKRs makes it easy to find and inspect knowledge-based applications and their components. Figure 4.18, Figure 5.16 and Figure 6.26 show how an EKR can be annotated using the PPR classes. Furthermore, the combination of a web-based knowledge management application with the PPR paradigm (as shown in Figure 6.27) has meant that EKRs, the contained knowledge, processes and cases are all easily accessible. Figure 5.18 shows the user process for cost model composition, which gives insight into the process steps that are necessary for finding and inspecting knowledge elements of an EKR using this knowledge management application.

Besides maintainability, knowledge change poses challenges on usability. Task orientation and expert / end user involvement are two specific challenges that the case studies have addressed. With regard to the challenge of *task orientation*, the EKR concept has been adopted as part of the KLC ontology to support the modelling, implementation and execution of specific engineering tasks within knowledge-based systems. EKRs contain the input, process and output for engineering tasks in the form of knowledge elements, process elements and cases. In Sections 4.2.2, 5.2.2 and 6.2.3, functional analysis has been performed to break down the specific engineering tasks (ply continuity optimization, cost modelling and estimation, slat assembly main downstop modification and inspection) into specific process activities. In each case study, one or more EKRs have been developed to represent and support the execution of these tasks.

The final challenge is *expert / end user involvement*. The case studies have shown that the use of a web-based knowledge management solution has the potential to involve the user in execution and control of engineering tasks. The best examples of user involvement in task execution are given in the manufacturing and maintenance case studies. For the manufacturing case study, the user process as given in Figure 5.18 shows how users can compose a cost model by inspecting, collating and downloading EKRs. Users can subsequently interact with the cost model in Excel. For the maintenance case study, users can

consult documentation and enter case reports online while executing the maintenance task. User involvement in task control has been shown in the manufacturing case study: the Excel cost model can be cross-checked with the knowledge base. Using the knowledge solution's validation cycle for knowledge elements, users can work on the basis of the most up-to-date knowledge. Finally, the solution devised for the design case study featured the most automation of steps. In theory, users can select the specific design and manufacturing constraints that they want to apply for the optimization of ply continuity. In practice, all constraints are automatically included into the analysis. In short, the developed case study solutions have the functionality to involve users in task execution and control. However, a clear downside of the proof-of-concept status of the solutions developed in the case studies is that the involvement of experts and end users has not been experimentally validated, though formal design reviews have been held.

In summary, three main academic contributions have been made in this dissertation:

1) **Knowledge Lifecycle model:** the model allows for the characterisation and quantification of knowledge change. It has been applied successfully for all case studies on a qualitative basis and the aircraft maintenance case study has shown that the model has additional potential to quantify knowledge change.
2) **Knowledge Lifecycle (KLC) ontology**: the KLC ontology leverages existing ontologies, the Enterprise Knowledge Resource concept, the Knowledge Lifecycle model and the Product-Process-Resource paradigm into one consistent model for supporting the development of knowledge-based applications that can cope with knowledge change. It has been applied in multiple aircraft lifecycle stages.
3) **Development of KNOMAD methodology:** the KNOMAD methodology has been introduced to support the development of knowledge-based applications for engineering tasks that are subject to knowledge change.

These academic contributions have been validated through a case study approach. It has been shown that the Knowledge Lifecycle Model, KNOMAD methodology and KLC ontology can successfully be applied in each aircraft lifecycle phase.

7.2 Research Conclusions

The vision of this research has been to show that knowledge changes and has a lifecycle which can be modelled and quantified, and to carry through the

implications of knowledge change into an ontology-based approach and a methodology to consistently formalize, use and maintain knowledge within the aircraft lifecycle. The following associated high-level research goal has been identified:

> *Support consistent formalization, use and maintenance of changing knowledge within aircraft lifecycle phases to improve domain-specific modelling, execution and control of engineering tasks*

This statement can be broken down into four main elements:

- **Support consistent formalization, use and maintenance [of]:** the KNOMAD methodology provides a consistent approach towards the formalization of knowledge. The KLC ontology has been set up to enable consistent use and maintenance of knowledge through use of the Enterprise Knowledge Resource concept and the Product-Process-Resource paradigm.
- **Changing knowledge:** the Knowledge Lifecycle Model has been developed to enable characterisation and measurement of knowledge change over time.
- **Within aircraft lifecycle phases:** case studies have been performed in the aircraft design, manufacturing and maintenance domains.
- **To improve domain-specific modelling, execution and control of engineering tasks:** the solutions developed for the case studies have improved modelling, execution and control of specific engineering tasks, as discussed in Sections 4.4, 5.4, 6.4 and the research synthesis (Section 7.1.2).

7.2.1 Theory Development: Knowledge Lifecycle Modelling

The first research objective is *knowledge lifecycle modelling*. The associated research questions are

- Which concepts and relationships are required to characterise the change of explicit knowledge within and throughout the aircraft lifecycle phases?
- How does explicit knowledge change within specific phases of the aircraft lifecycle?
- Is change of explicit knowledge quantifiable?

With respect to the first research question, the Knowledge Lifecycle model has been developed in Section 3.1 to allow for the characterisation and quantification of knowledge change. The model uses the concepts of *knowledge states* and *knowledge actions* to achieve this. In particular, knowledge actions (*create, formalize, use, maintain, update, retire*) can be used to quantify knowledge change.

With respect to the second research question, the model has been qualitatively applied for three specific phases of the aircraft lifecycle: design, manufacturing and maintenance. The knowledge states have been used to characterise knowledge change relative to each case study for these lifecycle phases.

The third research question has been addressed by quantifying the change of explicit knowledge for the aircraft maintenance domain on the subject of Airworthiness Directives (ADs) for two aircraft types: the Airbus A320 and the Boeing B737. The knowledge actions *create*, *maintain*, *update* and *retire* have been quantified for these samples. Within these boundaries, the knowledge lifecycle model has provided an adequate way of quantifying knowledge change over the lifecycle of an aircraft. From the samples, it has been possible to conclude that the frequency of knowledge actions decreases along the knowledge lifecycle. With respect to case-specific hypotheses, the main finding has been that a significant, small-to-medium size increasing trend in knowledge change per year can be observed from the A320 and B737 samples (R_{A320} = .497, $p_{A320} < .05$, R_{B767} = .644, $p_{B737} < .01$).

7.2.2 Theory Development: Ontology-based Approach to Support Knowledge Change

The second research objective is to develop an ontology-based approach to support knowledge change in knowledge-based applications, with a view to improved usability and maintainability of these applications. The associated research questions are

- Which concepts and mechanisms support the consistent formalization, use and maintenance of changing knowledge throughout the aircraft lifecycle?
- How can knowledge change be accommodated during knowledge-based application development?
    - Which models are required and how do these models help to accommodate knowledge change?

An ontology has been developed to support knowledge change in knowledge-based applications. This *Knowledge Life Cycle (KLC) Ontology* has been built using a number of contributing elements, which together address the first research question. The Product-Process-Resource paradigm used in Dassaults Systemes' PLM software is a main contribution and is used to represent the context in which engineering tasks are performed. The Enterprise Knowledge Resource concept is another essential contribution. EKRs are used to represent engineering tasks. An EKR is built up in a modular way and contains inputs, process and outputs in the form of knowledge elements, process elements and cases. Concepts and relationships included in previous ontologies such as the PROMISE Semantic

Object Model and the Core Ontology for Process Data Warehouse have been added into the KLC ontology. Finally, the knowledge lifecycle model introduced in Chapter 4 has been used to identify attributes that help in identifying and measuring knowledge change.

All contributions have been incorporated into a single ontology. The ontology has been modelled in Unified Modelling Language (UML) and is given in Figure 3.7. The ontology can be used to model and implement engineering tasks through the EKR concept; this contains the knowledge and process elements necessary for an engineering task and stores the output of the engineering task as case reports. The engineering task (as embodied in an EKR) can be semantically annotated through the Product, Process and Resource classes and subclasses, as shown in Figure 4.18, Figure 5.16, Figure 5.19 and Figure 6.26. The task and domain ontologies encompassed in the KLC ontology have been implemented in the AKM tool in the case studies, serving as the backbone for solution development and substantiating the ontology-based approach.

To answer the pair of research questions related to the accommodation of knowledge change, the ontology-based approach has been successfully employed in three case studies, addressing the design, manufacturing and maintenance aircraft lifecycle phases. In conjunction with the KNOMAD methodology, the KLC ontology has proven viable for KBS development in each lifecycle phase. For each phase, a knowledge-based application has been developed that addresses challenges related to the usability and maintainability of knowledge. In particular, the issues of transparency, 'black-box' applications, task orientation and user involvement have been addressed, as discussed in the research synthesis (Section 7.1) and the individual case study discussions (Sections 4.4, 5.4 and 6.4).

7.2.3 Theory Development: Methodology Development

The third research objective has been to develop a methodology for supporting the development of 'white-box' knowledge-based applications that can cope with knowledge change. The associated research questions are

- How can knowledge change be accommodated during knowledge-based application development?
    - Which steps are required?

To support the ontology-based approach towards knowledge-based application development, the KNOMAD methodology has been proposed in this dissertation. After justification of the business case for knowledge-based application development, the KNOMAD methodology contains steps for the capture of knowledge and subsequent identification of knowledge change. To this end, the Knowledge Lifecycle can be used to characterise and quantify knowledge change. Capture of knowledge is followed by normalization to comply with

(quality) standards. The methodology furthermore advises to develop domain-specific ontologies for the structuring and annotation of knowledge-based applications. These applications have to be modelled and implemented, followed by performance analysis and deployment into engineering practice. As such, the KNOMAD methodology answers the research questions mentioned above.

As mentioned before, the KNOMAD methodology has been successfully applied in conjunction with the KLC ontology in the development of knowledge-based applications for the design, manufacturing and maintenance domains. Usability and maintainability challenges have been addressed, as mentioned at the end of the previous section.

7.3 Research Limitations & Recommendations

In this Section, two sets of limitations are discussed: foreseen limitations due to the formulated research objectives, scope and design, and unforeseen limitations that have come up while performing research.

Most of the pre-existing limitations have been discussed in the research setup (Section 1.2.1.2) at some length. Interoperability of applications through aircraft life, knowledge exchange across aircraft lifecycle stages, organizational factors, automatic translation between informal and formal knowledge representation and automation are all considered as outside of scope, which translates to limitations on the breadth and implications of the performed research. With respect to the interoperability and organizational limitations, the research papers referenced in Sections 2.1, 2.2.2, 3.1.1 and 3.2.1 are good starting points for further research. The most potential for future research is related to the automatic translation between informal and formal knowledge representations. A potentially much more elegant solution to account for knowledge change would be to have coupled informal and formal models, including code generation & implementation (Verhagen *et al.*, 2012). Knowledge can be represented in informal terms (for end users) in the knowledge base and this can be translated automatically into detailed, KBE application-specific language. End users can work directly on the basis of informal knowledge – any changes made by them are incorporated automatically into the code. Vice versa, any changes to the KBE application can be incorporated in the knowledge base – the knowledge base and KBE application are fully synchronized. The iPROD European Seventh Framework Programme project (iProd, 2013) is working towards this goal, as is research in the Flight Performance and Propulsion chair at Delft University of Technology (Van Dijk *et al.*, 2012; Chan, 2013).

Another research limitation derives from the research design. The choice for practice-oriented case study research means that results from these case studies cannot be straightforwardly generalized. The generalization of the ontology-based

approach to KBS development must be validated through more case studies and in industrial settings (use 'in anger').

The following research limitations and associated recommendations have emerged during and from the research performed and discussed in this dissertation.

1) Knowledge lifecycle modelling & validation

Limitation: as mentioned before, no quantification of knowledge change has been performed for the aircraft design and manufacturing domains. Knowledge change has been discussed on a case study basis. However, the Knowledge Lifecycle Model evidently has to be quantitatively validated across more domains and (potentially) products to note its strengths, shortcomings and implications and thereby to gain acceptance in the scientific community.

Recommendation: establishing a formal foundation for the Knowledge Lifecycle model would aid quantification, analysis and replication of results. Investigation and application of TMS (Doyle, 1979; Katsuno and Mendelzon, 1991) to quantify knowledge change in propositional knowledge bases is of particular promise. The Knowledge Lifecycle model has to be validated across more domains and product types.

2) Task complexity due to multiple element interactions

Limitation: The modular approach to development of KBS as expressed in the KLC ontology (e.g. reflected in the use of **Knowledge_Element** and **Process_Element** classes) poses a potential problem: the number of (potential) interfaces between these elements grow quickly, leading to increasingly complex systems. Is there a natural limit; how many elements should an EKR consist of? The complexity resulting from system element interactions is briefly explored in Appendix A: Complexity Estimation.

Recommendation: It is necessary to further investigate this issue and if possible, empirically assess whether and where limits on the use of modular elements can be found. This can result in guidelines for the use of a modular approach to KBS development.

3) Verification and validation

Limitation: Verification can be defined by considering the question: “Are we building the product right?” (Boehm, 1981). Alternatively, it can be seen as the 'process of testing that a product meets its specification' (Coenen and Bench-Capon, 1993).

Validation refers to “Are we building the right product?” (Boehm, 1981) and can be seen as 'the process of testing that a product satisfies the requirements of the customer' (Coenen and Bench-Capon, 1993). The three case studies provide validation of the ontology-based approach towards KBS development. However, verification is not explicitly taken into account: there is no proof to the claim that the ontology-based approach of building KBS using KNOMAD and the KLC ontology is the 'right' approach. In fact, alternative approaches may be just as good or even better: see also the point made previously with respect to automatic translation between informal and formal representations of knowledge. However, the simplicity, low conceptual entry barrier and wide applicability of the ontology-based approach are potential advantages.

Recommendation: it is recommended to conduct a case study using multiple approaches and evaluate the results using predetermined indicators. Confounding variables such as a learning effect and case study team composition must be taken into account and neutralized.

4) Process maturity evaluation

Limitation: evaluating the maturity and stability of a process may inform decision-making about following a 'static' or 'dynamic' approach (see also Figure 7.1).

Recommendation: The Knowledge Lifecycle model can potentially be part of the maturity assessment. Research must establish how the concepts from this model can be fruitfully applied in a process maturity assessment.

5) Beyond the current ontology-based approach to knowledge change in knowledge-based applications:

i. **Task hierarchy:** how does the ontology-based approach scale up when a hierarchy of tasks must be modelled and represented? Can EKRs be stacked onto each other? This issue is not investigated here; a more formal in-depth study on task modelling and implementation is necessary, for instance building upon the research related to the IDEF modelling technique (Integrated Definition Methods, 2012) and the Function-Behaviour-State theory (Umeda *et al.*, 1990; Umeda *et al.*, 1995) and translating that towards the ontology-based approach.

i. **Making use of ontology capabilities:** the KLC ontology is currently not expressed in a formal way, e.g. through the use

of first order logic (FOL) predicates. Doing so would aid academically and practically: from an academic perspective, it becomes easier to validate the semantics of the ontology. With that, reproducibility and criticism of the ontology is facilitated. From a practical perspective, the reasoning capabilities associated with a formal ontology – e.g. subsumption, coherence, identity, compatability – can be facilitated and used in the development and maintenance of knowledge-based systems.

This research may serve as the starting point for several avenues of further research, such as recommended above. It is hoped that the consideration of a broad scope – including three quite different phases of the aircraft lifecycle – may lead to follow-up research regarding the issues surrounding knowledge change. Other scholars are invited to refine, revise, refute and/or expand the insights that have been developed in this dissertation.

References

Abramovici, M. (2007). "Future Trends in Product Lifecycle Management (PLM): The Future of Product Development". F.-L. Krause (eds.), Springer Berlin Heidelberg, pp. 665-674.

Airbus. (2011). "Global Market Forecast 2010 - 2029." Retrieved 16-08-2011, from http://www.airbus.com/company/market/gmf2010/.

Airbus. (2012). "AIRMAN." Retrieved 13-11-2012, from http://www.airbus.com/innovation/proven-concepts/in-fleet-support/airman/.

Aitken, J., P. Childerhouse and D. Towill (2003). "The impact of product life cycle on supply chain strategy". *International Journal of Production Economics*, **85**(2): pp. 127-140.

Alavi, M. and D. E. Leidner (2001). "Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues". *Mis Quarterly*, **25**(1): pp. 107-136.

Ameri, F. and D. Dutta (2005). "Product lifecycle management: Closing the knowledge loops". *Computer-Aided Design and Applications*, **2**(5): pp. 577-590.

Amodio, C. C., C. Cziulik, C. Ugaya, E. Fernandes, F. Siqueira, H. Rozenfeld, . . . S. Branício (2008). "Ontologia PLM Project: Development and Preliminary Results". R. Curran, S.-Y. Chou and A. Trappey (eds.), *Collaborative Product and Service Life Cycle Management for a Sustainable World*. London, Springer, pp. 503-511.

Aviation-Week. (2011). "Airbus Refines A30X Design." Retrieved 15-08-2011, from http://www.aviationweek.com/aw/generic/story_generic.jsp?channel=awst&id=news/aw020909p3.xml&headline=null&next=0.

Baker, A., S. Dutton and D. Kelly (2004). *Composite Materials for Aircraft Structures*, Reston, Virginia, American Institute of Aeronautics and Astronautics.

Baker, M., T. Dowling, W. Martinez, T. Medejski, D. Pedersen and D. Rockwell (2006). "New Enhanced Service Bulletins". *Aero Quarterly*, **4**(6): pp. 12-15.

Bermell-Garcia, P. (2007). "A metamodel to annotate knowledge based engineering codes as enterprise knowledge resources". Cranfield University, Ph.D. dissertation.

Bermell-Garcia, P. and I. S. Fan (2008). "Practitioner requirements for integrated Knowledge-Based Engineering in Product Lifecycle Management". *International Journal of Product Lifecycle Management*, **3**(1): pp. 3-20.

Bermell-Garcia, P., W. J. C. Verhagen, S. Astwood, K. Krishnamurthy, J. L. Johnson, D. Ruiz, . . . R. Curran (2012). "A framework for management of Knowledge-Based Engineering applications as software services: Enabling personalization and codification". *Advanced Engineering Informatics*, **26**(2): pp. 219-230.

Bilgic, T. and D. Rock (1997). "Product Data Management Systems: State of the Art and the Future". *DETC '97; 1997 ASME Design Engineering Technical Conferences*, Sacramento, California, ASME.

Birkinshaw, J. and T. Sheehan (2002). "Managing the knowledge life cycle". *MIT Sloan Management Review*, **44**(1): pp. 75-83.

Blom, A. W. (2010). "Structural Performance of Fiber-Placed, Variable-Stiffness Composite Conical and Cylindrical Shells". Delft University of Technology, Ph.D. dissertation.

Boehm, B. W. (1981). *Software Engineering Economics*, Prentice-Hall.

Boeing (2010). Boeing Service Bulletin 737-57A1302, Revision 1.

Borgo, S. and P. Leitão (2007). "Foundations for a Core Ontology of Manufacturing Ontologies". R. Sharman, R. Kishore and R. Ramesh (eds.), *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, Springer, pp. 751-775.

Borst, P., H. Akkermans and J. Top (1997). "Engineering ontologies". *International Journal of Human Computer Studies*, **46**(2-3): pp. 365-406.

Brandt, S. C., J. Morbach, M. Miatidis, M. Theißen, M. Jarke and W. Marquardt (2008). "An ontology-based approach to knowledge management in design processes". *Computers and Chemical Engineering*, **32**(1-2): pp. 320-342.

Brissaud, D. and S. Tichkiewitch (2001). "Product Models for Life-Cycle". *CIRP Annals - Manufacturing Technology*, **50**(1): pp. 105-108.

Brockmans, S., R. Volz, A. Eberhart and P. Löffler (2004). "Visual modeling of OWL DL ontologies using UML". *The Semantic Web–ISWC 2004*: pp. 198-213.

Bruyneel, M. (2011). "SFP-a new parameterization based on shape functions for optimal material selection: Application to conventional composite plies". *Structural and Multidisciplinary Optimization*, **43**(1): pp. 17-27.

Bruyneel, M., C. Beghin, G. Craveur and F. Colsoul (2012). "Topology optimization of composite structures with continuous design variables". *2012 SAMPE International Symposium and Exhibition - Emerging Opportunities: Materials and Process Solutions*, Baltimore, MD.

Buckowitz, W. R. and R. L. Williams (1999). *Knowledge management fieldbook*, London, Prentice-Hall.

Bufardi, A., P. Folan and K. Cormican (2005) "DR7.1: Concepts for translation and transformation of information to knowledge". PROMISE Consortium 2004 - 2008. 507100 PROMISE; A Project of the 6th Framework Programme Information Society Technologies (IST).

Burhani, S. (2012). "Compliance during Aircraft (Component) Redeliveries". Delft University of Technology, M.Sc. thesis.

Butterfield, J., W. McEwan, P. Han, M. Price, D. Soban and A. Murphy (2012). "Digital Methods for Process Development in Manufacturing and Their Relevance to Value Driven Design". *Air Transport and Operations - Proceedings of the Second International Air Transport and Operations Symposium 2011*, Delft, The Netherlands, IOS Press.

Chan, P. K. M. (2013). "A New Methodology for the Development of Simulation Workflows". Delft University of Technology, M.Sc. thesis.

Choi, J. W., D. Kelly, J. Raju and C. Reidsema (2005). "Knowledge-based engineering system to estimate manufacturing cost for composite structures". *Journal of Aircraft*, **42**(6): pp. 1396-1402.

Coenen, F. and T. Bench-Capon (1993). *Maintenance of Knowledge-Based Systems: Theory, Techniques and Tools*, London, Academic Press.

Colledani, M., W. Terkaj, T. Tollio and M. Tomasella (2008). "Development of a conceptual reference framework to manage manufacturing knowledge related to products, processes and production systems". A. Bernard and S. Tichkiewitch (eds.), *Methods and tools for effective knowledge life-cycle management*. Berlin, Springer, pp. 3-21.

Compositesworld. (2011). "Thermoplastic composites on tap for the A30X, new process in testing " Retrieved 15-08-2011, from http://www.compositesworld.com/news/afp-demonstrator-produces-thermoplastic-composite-parts-for-airbus-a30x.

Cooper, C. A. (2011). "Development of a Methodology to Support Design of Complex Aircraft Wings". Delft University of Technology, Ph.D. dissertation.

Cooper, D. and G. La Rocca (2007). "Knowledge-based Techniques for Developing Engineering Applications in the 21st Century". *7th AIAA ATIO Conference*, Belfast, Northern Ireland, AIAA 2007-7711.

Corallo, A., R. Laubacher, A. Margherita and G. Turrisi (2009). "Enhancing product development through knowledge-based engineering (KBE): A case study in the aerospace industry". *Journal of Manufacturing Technology Management*, **20**(8): pp. 1070 - 1083.

Cranefield, S. and M. Purvis (1999). "UML as an Ontology Modeling Language". *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on AI (IJCAI-99)*, Stockholm.

Curran, R., J. Butterfield, Y. Jin, R. Collins and R. Burke (2010). "Value-Driven Manufacture: Digital Lean Manufacture". R. Blockley and W. Shyy (eds.), *Encyclopedia of Aerospace Engineering*, John Wiley & Sons, Ltd.

Curran, R., S. Raghunathan and M. Price (2004). "Review of aerospace engineering cost modelling: The genetic causal approach". *Progress in Aerospace Sciences*, **40**(8): pp. 487-534.

Curran, R., W. J. C. Verhagen, M. J. L. Van Tooren and A. H. Van der Laan (2010). "A multidisciplinary implementation methodology for knowledge based engineering: KNOMAD". *Expert Systems with Applications*, **37**(11): pp. 7336-7350.

Dassault Systemes. (2012). "CATIA V5R16 - Fact Sheet." Retrieved 16-12-2012, from http://www.3ds.com/fileadmin/V5R16/CATIA_V5R16_Factsheets_final.pdf.

Davenport, T. H. and L. Prusak (1998). *Working Knowledge: How Organisations Manage What They Know*, Boston, Harvard Business Press.

de Kleer, J. (1986). "An assumption-based TMS". *Artificial Intelligence*, **28**(2): pp. 127-162.

Doyle, J. (1979). "A truth maintenance system". *Artificial Intelligence*, **12**(3): pp. 231-272.

EASA. (2012). "EASA Airworthiness Directives Publishing Tool." Retrieved 14th February, from http://ad.easa.europa.eu/.

Elgh, F. and M. Cederfeldt (2010). "Documentation and Management of Product Knowledge in a System for Automated Variant Design: A Case Study". J. Pokojski, S. Fukuda and J. Salwiński (eds.), *New World Situation: New Directions in Concurrent Engineering*, Springer London, pp. 237-245.

Emberey, C. L., N. Milton, J. P. T. J. Berends, M. J. L. Van Tooren, S. W. G. Van der Elst and B. Vermeulen (2007). "Application of Knowledge Engineering Methodologies to Support Engineering Design Application Development in Aerospace". *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland.

Epistemics. "PCPACK." Retrieved 17-01-2011, from http://www.epistemics.co.uk/Notes/55-0-0.htm.

Er, A. and R. Dias (2000). "A rule-based expert system approach to process selection for cast components". *Knowledge-Based Systems*, **13**(4): pp. 225-234.

Erden, M., H. Komoto, T. Van Beek, V. D'amelio, E. Echavarria and T. Tomiyama (2008). "A review of function modeling: Approaches and applications". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **22**(02): pp. 147-169.

European Federation of National Maintenance Societies. (2011). "What does EFNMS stand for?" Retrieved 15 December, from http://www.efnms.org/What-EFNMS-stands-for/m13l2/What-EFNMS-stands-for.html.

Eynard, B., T. Gallet, P. Nowak and L. Roucoules (2004). "UML based specifications of PDM product structure and workflow". *Computers in Industry*, **55**(3): pp. 301-316.

Eynard, B., T. Gallet, L. Roucoules and G. Ducellier (2006). "PDM system implementation based on UML". *Mathematics and Computers in Simulation*, **70**(5-6): pp. 330-342.

FAA (2007). Airworthiness Directive FAA AD 2007-18-52.

FAA (2011). Airworthiness Directive FAA AD 2011-06-05.

FAA. (2012). "RGL - Airworthiness Directives." Retrieved 13-06-2012, from http://www.airweb.faa.gov/Regulatory_and_Guidance_Library/rgAD.nsf/MainFrame?OpenFrameSet

Fan, I. S. and P. Bermell-Garcia (2008). "International Standard Development for Knowledge Based Engineering Services for Product Lifecycle Management". *Concurrent Engineering-Research and Applications*, **16**(4): pp. 271-277.

Fan, I. S., G. Li, M. Lagos-Hernandez, P. Bermell-Garcia and M. Twelves (2002). "A Rule Level Knowledge Management System for Knowledge Based Engineering Applications". *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2002)*, Montreal, Quebec, Canada ASME.

Feigenbaum, E. A. and P. McCorduck (1983). *The fifth generation : artificial intelligence and Japan's computer challenge to the world*, Reading, Mass., USA, Addison-Wesley.

Feldman, P. and A. Shtub (2006). "Model for cost estimation in a finite-capacity environment". *International Journal of Production Research*, **44**(2): pp. 305 - 327.

Fernandes Lopez, M., A. Gomez Perez and N. Juristo (1997). "METHONTOLOGY: From Ontological Art Towards Ontological Engineering". *AAAI-97 Spring Symposium Series*, Stanford.

Främling, K. and L. Rabe (2005) "DR 7.2: Concept and methods for information enrichment". PROMISE Consortium 2004 - 2008. 507100 PROMISE; A Project of the 6th Framework Programme Information Society Technologies (IST).

Gärdenfors, P. (2003). *Belief revision*, Cambridge University Press.

Garg, A. and S. G. Deshmukh (2006). "Maintenance management: literature review and directions". *Journal of Quality in Maintenance Engineering*, **12**(3): pp. 205 - 238.

Geddes, N. and R. Armstrong (1991). "Knowledge maintenance in an evolving system using a deep structure representation". *AIAA Computing in Aerospace Conference*, Baltimore, MD, USA.

Gielingh, W. F. (2005). "Improving the Performance of Construction by Acquisition, Organisation and Use of Knowledge". Delft University of Technology, Ph.D. dissertation.

Gillet, A., P. Francescato and P. Saffre (2010). "Single- and multi-objective optimization of composite structures: The influence of design variables". *Journal of Composite Materials*, **44**(4): pp. 457-480.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications". *Knowledge Acquisition*, **5**(2): pp. 199-220.

Gunawan, S., S. Azarm, J. Wu and A. Boyars (2003). "Quality-assisted multi-objective multidisciplinary genetic algorithms". *Aiaa Journal*, **41**(9): pp. 1752-1762.

Hicks, B. J., S. J. Culley, R. D. Allen and G. Mullineux (2002). "A framework for the requirements of capturing, storing and reusing information and knowledge in engineering design". *International Journal of Information Management*, **22**(4): pp. 263-280.

IJsselmuiden, S. T., M. M. Abdalla, O. Seresta and Z. Gürdal (2009). "Multi-step blended stacking sequence design of panel assemblies with buckling constraints". *Composites Part B: Engineering*, **40**(4): pp. 329-336.

Integrated Definition Methods. (2012). "IDEF0 - Function Modeling Method." Retrieved 16-12-2012, from http://www.idef.com/IDEF0.htm.

iProd. (2013). "Improving the Product Development Process (PDP - iPROD project)." Retrieved 20-03-2013, from http://www.iprod-project.eu/index.

Jagtap, S. and A. Johnson (2011). "In-service information required by engineering designers". *Research in Engineering Design*, **22**(4): pp. 207-221.

Jin, P., B. Song and X. Zhong (2011). "Structure optimization of large composite wing box with parallel genetic algorithm". *Journal of Aircraft*, **48**(6): pp. 2145-2148.

Jun, H. B., D. Kiritsis and P. Xirouchakis (2007). "Research issues on closed-loop PLM". *Computers in Industry*, **58**(8-9): pp. 855-868.

Katsuno, H. and A. O. Mendelzon (1991). "Propositional knowledge base revision and minimal change". *Artificial Intelligence*, **52**(3): pp. 263-294.

Kern-Isberner, G. (2004). "A Thorough Axiomatization of a Principle of Conditional Preservation in Belief Revision". *Annals of Mathematics and Artificial Intelligence*, **40**(1-2): pp. 127-164.

Kiritsis, D., A. Bufardi and P. Xirouchakis (2003). "Research issues on product lifecycle management and information tracking using smart embedded systems". *Advanced Engineering Informatics*, **17**(3-4): pp. 189-202.

Kitamura, Y., M. Kashiwase, M. Fuse and R. Mizoguchi (2004). "Deployment of an ontological framework of functional design knowledge". *Advanced Engineering Informatics*, **18**(2): pp. 115-127.

Ko, K. H., K. Pochiraju and S. Manoochehri (2007). "An embedded system for knowledge-based cost evaluation of molded parts". *Knowledge-Based Systems*, **20**(3): pp. 291-299.

Kristinsdottir, B. P., Z. B. Zabinsky, M. E. Tuttle and S. Neogi (2001). "Optimal design of large composite panels with varying loads". *Composite Structures*, **51**(1): pp. 93-102.

Krozer, Y. (2008). "Life cycle costing for innovations in product chains". *Journal of Cleaner Production*, **16**(3): pp. 310-321.

Kuhn, O. (2010). "Methodology for Knowledge-Based Engineering Update". L'Université Claude Bernard Lyon, Ph.D. dissertation.

Kulon, J., D. J. Mynors and P. Broomhead (2006). "A knowledge-based engineering design tool for metal forging". *Journal of Materials Processing Technology*, **177**(1-3): pp. 331-335.

La Rocca, G. (2011). "Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization". Delft University of Technology, Ph.D. dissertation.

La Rocca, G. (2011). "Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization". Delft University of Technology, Ph.D. dissertation.

La Rocca, G. (2012). "Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design". *Advanced Engineering Informatics*, **26**(2): pp. 159-179.

La Rocca, G. and M. J. L. van Tooren (2009). "Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization". *Journal of Aircraft*, **46**(6): pp. 1875-1885.

Lampe, M., M. Strassner and E. Fleisch (2004). "A Ubiquitous Computing environment for aircraft maintenance". *Proceedings of the 2004 ACM symposium on Applied computing*, Nicosia, Cyprus, ACM.

Lee, J. H. and H. W. Suh (2008). "Ontology-based multi-layered knowledge framework for product lifecycle management". *Concurrent Engineering Research and Applications*, **16**(4): pp. 301-311.

Lee, S. G., Y. S. Ma, G. L. Thimm and J. Verstraeten (2008). "Product lifecycle management in aviation maintenance, repair and overhaul". *Computers in Industry*, **59**(2-3): pp. 296-303.

Liu, B. and R. T. Haftka (2004). "Single-level composite wing optimization based on flexural lamination parameters". *Structural and Multidisciplinary Optimization*, **26**(1-2): pp. 111-120.

Liu, B., R. T. Haftka and M. A. Akgün (2000). "Two-level composite wing structural optimization using response surfaces". *Structural and Multidisciplinary Optimization*, **20**(2): pp. 87-96.

Liu, D., V. V. Toropov, O. M. Querin and D. C. Barton (2011). "Bilevel optimization of blended composite wing panels". *Journal of Aircraft*, **48**(1): pp. 107-118.

Liu, D., V. V. Toropov, M. Zhou, D. C. Barton and O. M. Querin (2010). "Optimization of blended composite wing panels using smeared stiffness technique and lamination parameters".

Liu, T. D. and W. X. Xu (2001). "A review of web-based product data management systems". *Computers in Industry*, **44**(3): pp. 251-262.

Lovett, P. J., A. Ingram and C. N. Bancroft (2000). "Knowledge-based engineering for SMEs -- a methodology". *Journal of Materials Processing Technology*, **107**(1-3): pp. 384-389.

Ma, Q. C. and X. W. Liu (2007). "Review of Knowledge Based Engineering with PLM". *Applied Mechanics and Materials*, **10-12**(-): pp. 127-131.

Maksimovic, M., A. Al-Ashaab, E. Shehab and R. Sulowski (2011). "A Lean Knowledge Lifecycle Methodology in Product Development". *8th International Conference on Intellectual Capital, Knowledge Management & Organisational Learning*, Bangkok, Thailand, Academic Publishing Limited.

Markus, M. L. (2001). "Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success". *Journal of Management Information Systems*, **18**(1): pp. 57-93.

Maropoulos, P. (2003). "Digital enterprise technology--defining perspectives and research priorities". *International Journal of Computer Integrated Manufacturing*, **16**(7-8): pp. 467-478.

Martins, J. P. (1990). "The truth, the whole truth, and nothing but the truth". *AI Magazine*, **11**(4): pp. 7.

Martins, J. P. and S. C. Shapiro (1988). "A model for belief revision". *Artificial Intelligence*, **35**(1): pp. 25-79.
Matsokis, A. (2010). "An Ontology-Based Approach for Closed-Loop Product Lifecycle Management". École Polytechnique Federale de Lausanne, Ph.D. dissertation.
Matsokis, A. and D. Kiritsis (2010). "An ontology-based approach for Product Lifecycle Management". *Computers in Industry*, **61**(8): pp. 787-797.
Mazumdar, S. K. (2002). *Composites Manufacturing: Materials, Product, and Process Engineering*, Boca Raton, Florida, CRC Press.
McDermott, D. and J. Doyle (1980). "Non-monotonic logic I". *Artificial Intelligence*, **13**(1-2): pp. 41-72.
McDonough, W. and M. Braungart (2002). *Cradle to Cradle: Remaking The Way We Make Things*, New York, North Point Press.
McElroy, M. W. (2003). *The new knowledge management*, Boston, MA, KMCI Press, Butterworth-Heinemann.
McMahon, C., M. Giess and S. Culley (2005). "Information management for through life product support: The curation of digital engineering data". *International Journal of Product Lifecycle Management*, **1**(1): pp. 26-42.
McMahon, C., A. Lowe and S. Culley (2004). "Knowledge management in engineering design: Personalization and codification". *Journal of Engineering Design*, **15**(4): pp. 307-325.
McQueen, R. (1998). "Four Views of Knowledge and Knowledge Management". *Proceedings of the Fourth Americas Conference on Information Systems*.
Merali, Y. and J. Davies (2001). "Knowledge capture and utilization in virtual communities". *Proceedings of the 1st International Conference on Knowledge Capture (K-CAP)*, Victoria, BC, Canada, ACM.
Milton, N. R. (2007). *Knowledge acquisition in practice: a step-by-step guide*, Springer-Verlag London Limited.
Mizoguchi, R., Y. Tijerino and M. Ikeda (1995). "Task analysis interview based on task ontology". *Expert Systems with Applications*, **9**(1): pp. 15-25.
Mohan, K. and B. Ramesh (2007). "Traceability-based knowledge integration in group decision and negotiation activities". *Decision Support Systems*, **43**(3): pp. 968-989.
National Institute of Standards and Technology (1993). Integration Definition for Function Modelling (IDEF0), Federal Information Processing Standards Publication 183.
Newman, B. D. and K. W. Conrad (2000). "A Framework for Characterizing Knowledge Management Methods, Practices, and Technologies". *Proceedings of PAKM*.
Newnes, L. B., A. R. Mileham, W. M. Cheung, R. Marsh, J. D. Lanham, M. E. Saravi and R. W. Bradbery (2008). "Predicting the whole-life cost of a product at the conceptual design stage". *Journal of Engineering Design*, **19**(2): pp. 99 - 112.
Niazi, A., J. S. Dai, S. Balabani and L. Seneviratne (2006). "Product Cost Estimation: Technique Classification and Methodology Review". *Journal of Manufacturing Science and Engineering*, **128**(2): pp. 563-575.
Nonaka, I. (1994). "A Dynamic Theory of Organizational Knowledge Creation". *Organization Science*, **5**(1): pp. 14-37.
Nonaka, I., R. Toyama and N. Konno (2000). "SECI, Ba and Leadership: A Unified Model of Dynamic Knowledge Creation". *Long Range Planning*, **33**(1): pp. 5-34.
Northwest-Aerospace-Alliance. (2010). "Airbus Next Generation Composite Wing." Retrieved 02 February, from http://www.aerospace.co.uk/technologies/ngcw.php.
Noy, N. F. and D. L. McGuinness (2009) "Ontology development 101: a guide to creating your first ontology". Stanford University.
Object Management Group. "UML." Retrieved 03-03-2013, from http://www.uml.org/.

Oldham, K., S. Kneebone, M. Callot, A. Murton and R. Brimble (1998). "MOKA - A methodology and tools oriented to knowledge-based engineering applications". *Changing the Ways We Work*, **8**: pp. 198-207.

Ouertani, M. Z., S. Baïna, L. Gzara and G. Morel (2011). "Traceability and management of dispersed product knowledge during design and manufacturing". *CAD Computer Aided Design*, **43**(5): pp. 546-562.

Panesar, A. S. and P. M. Weaver (2012). "Optimisation of blended bistable laminates for a morphing flap". *Composite Structures*, **94**(10): pp. 3092-3105.

Panetto, H., M. Dassisti and A. Tursi (2012). "ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment". *Advanced Engineering Informatics*, **26**(2): pp. 334-348.

Peak, R. S., J. Lubell, V. Srinivasan and S. C. Waterbury (2004). "STEP, XML, and UML: Complementary Technologies". *Journal of Computing and Information Science in Engineering*, **4**(4): pp. 379-390.

PEGASUS. (2013). "PEGASUS Project: Integrated engineering processing & materials technologies for the European sector." Retrieved 29-04-2013, from www.pegasus-eu.net.

Philpotts, M. (1996). "Introduction to the concepts, benefits and terminology of product data management". *Industrial Management and Data Systems*, **96**(4): pp. 11-17.

Pinto, H. S. and J. P. Martins (2004). "Ontologies: How can they be built?". *Knowledge and Information Systems*, **6**(4): pp. 441-464.

Polanyi, M. (1966). *The Tacit Dimension*, London, Routledge & Kegan Paul.

Preston, S., C. Chapman, M. Pinfold and G. Smith (2005). "Knowledge acquisition for knowledge-based engineering systems". *International Journal of Information Technology and Management*, **4**(1): pp. 1-11.

Price, M., S. Raghunathan and R. Curran (2006). "An integrated systems engineering approach to aircraft design". *Progress in Aerospace Sciences*, **42**(4): pp. 331-376.

Raymer, D. P. (2006). *Aircraft Design: A Conceptual Approach*. 4th, Reston, Virginia, American Institute of Aeronautics and Astronautics, Inc.

Rizzi, A., M. Zhang, B. Nagel, D. Boehnke and P. Saquet (2012). "Towards a unified framework using CPACS for geometry management in aircraft design".

Rodriguez, K. and A. Al-Ashaab (2007). "Knowledge web-based system to support e-manufacturing of injection moulded products". *International Journal of Manufacturing Technology and Management*, **10**(4): pp. 400-418.

Roy, U., N. Pramanik, R. Sudarsan, R. D. Sriram and K. W. Lyons (2001). "Function-to-form mapping: Model, representation and applications in design synthesis". *CAD Computer Aided Design*, **33**(10): pp. 699-719.

Sainter, P., K. Oldham and A. Larkin (2000). "Achieving benefits from Knowledge-Based Engineering systems in the longer term as well as in the short term". *Proceedings International Conference on Concurrent Enterprising*, Toulouse, France.

Schorlemmer, M., S. Potter, D. Robertson and D. Sleeman (2002). "Knowledge Life Cycle Management over a Distributed Architecture". *Expert Update*, **5**(3): pp. 2-19.

Schreiber, G., H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde and B. Wielinga (1999). *Knowledge engineering and management: the CommonKADS methodology*, Cambridge, MA, MIT Press.

Siemieniuch, C. E. and M. A. Sinclair (1999). "Organizational aspects of knowledge lifecycle management in manufacturing". *International Journal of Human-Computer Studies*, **51**(3): pp. 517-547.

Siemieniuch, C. E. and M. A. Sinclair (2004). "CLEVER: A process framework for knowledge lifecycle management". *International Journal of Operations and Production Management*, **24**(11): pp. 1104-1125.

Simon, M., G. Bee, P. Moore, J. S. Pu and C. Xie (2001). "Modelling of the life cycle of products with data acquisition features". *Computers in Industry*, **45**(2): pp. 111-122.

Soremekun, G., Z. Gürdal, C. Kassapoglou and D. Toni (2002). "Stacking sequence blending of multiple composite laminates using genetic algorithms". *Composite Structures*, **56**(1): pp. 53-62.

Stokes, M. (2001). *Managing Engineering Knowledge – MOKA: Methodology for Knowledge Based Engineering Applications*, London, Professional Engineering Publishing Limited.

Studer, R., V. R. Benjamins and D. Fensel (1998). "Knowledge Engineering: Principles and methods". *Data and Knowledge Engineering*, **25**(1-2): pp. 161-197.

Sudarsan, R., S. J. Fenves, R. D. Sriram and F. Wang (2005). "A product information modeling framework for product lifecycle management". *CAD Computer Aided Design*, **37**(13): pp. 1399-1411.

Sudarsan, R., S. J. Fenves, R. D. Sriram and F. Wang (2005). "A product information modeling framework for product lifecycle management". *Computer-Aided Design*, **37**(13): pp. 1399-1411.

Sunnersjo, S., M. Cederfeldt, F. Elgh and I. Rask (2006). "A Transparent Design System for Iterative Product Development". *Journal of Computing and Information Science in Engineering*, **6**(3): pp. 300-307.

Thimm, G., S. G. Lee and Y. S. Ma (2006). "Towards unified modelling of product life-cycles". *Computers in Industry*, **57**(4): pp. 331-341.

Tomasella, M., J. Cassina, A. Metin and M. Marquard (2006) "DR9.2: Specification of the System Object Model". PROMISE Consortium 2004 - 2008. 507100 PROMISE; A Project of the 6th Framework Programme Information Society Technologies (IST).

Transport Canada. (1996). "Canadian Aviation Regulations - Part V - Standard 593 Appendix A - Airworthiness Directive Format." Retrieved 27th February, from http://www.tc.gc.ca/eng/civilaviation/regserv/cars/part5-standards-a593sa-255.htm.

Transport Canada. (2002). "Canadian Aviation Regulations - Part V - Standard 593 - Airworthiness Directives." Retrieved 27th February, from http://www.tc.gc.ca/eng/civilaviation/regserv/cars/part5-standards-593s-1829.htm

Tsang, A. H. C. (1995). "Condition-based maintenance: Tools and decision making". *Journal of Quality in Maintenance Engineering*, **1**(3): pp. 3-17.

Tuomi, I. (1999). "Data is more than knowledge: implications of the reversed knowledge hierarchy for knowledge management and organizational memory". *HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, Hawaii, USA.

Umeda, Y., M. Ishii, M. Yoshioka, Y. Shimomura and T. Tomiyama (1996). "Supporting conceptual design based on the function-behavior-state modeler". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, **10**(4): pp. 275-288.

Umeda, Y., H. Takeda, T. Tomiyama and H. Yoshikawa (1990). "Function, behaviour, and structure". *Applications of artificial intelligence in engineering V*, **1**: pp. 177-194.

Umeda, Y., T. Tomiyama and H. Yoshikawa (1995). "FBS modeling: modeling scheme of function for conceptual design". *Proceedings of the 9th International Workshop on Qualitative Reasoning*, Amsterdam.

Uschold, M. (1996). "Building Ontologies: Towards a Unified Methodology". *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK.

Uschold, M. and M. Gruninger (1996). "Ontologies: Principles, methods and applications". *Knowledge Engineering Review*, **11**(2): pp. 93-136.

Usman, Z., R. Young, N. Chungoora, C. Palmer, K. Case and J. Harding (2011). "A Manufacturing Core Concepts Ontology for Product Lifecycle Interoperability". M. Sinderen and P. Johnson (eds.), *Enterprise Interoperability*, Springer Berlin Heidelberg, pp. 5-18.

Van der Elst, S. W. G. (2007). "Design of a Knowledge Based Prepreg Sub-Division Process". Delft University of Technology, M.Sc. thesis.

Van der Laan, A. H. (2008). "Knowledge based engineering support for aircraft component design". Delft University of Technology, Ph.D. dissertation.

Van der Spek, R. and A. Spijkervet (1997). *Knowledge management: dealing intelligently with knowledge*, New York, CRC Press.

Van der Velden, C., C. Bil and X. Xu (2012). "Adaptable methodology for automation application development". *Advanced Engineering Informatics*, **26**(2): pp. 231-250.

Van Dijk, R., X. Zhao, H. Wang and F. Van Dalen (2012). "Multidisciplinary Design and Optimization Framework for Aircraft Box Structures". *3rd Aircraft Structural Design Conference*, Delft.

Verhagen, W. J. C., P. Bermell-Garcia, P. Mariot, J.-P. Cotton, D. Ruiz, R. Redon and R. Curran (2012). "Knowledge-based cost modelling of composite wing structures". *International Journal of Computer Integrated Manufacturing*, **25**(4-5): pp. 368-383.

Verhagen, W. J. C., P. Bermell-Garcia, R. E. C. Van Dijk and R. Curran (2012). "A critical review of Knowledge-Based Engineering: An identification of research challenges". *Advanced Engineering Informatics*, **26**(1): pp. 5-15.

Verhagen, W. J. C. and R. Curran (2011). "Ontological Modelling of the Aerospace Composite Manufacturing Domain". D. D. Frey, S. Fukuda and G. Rock (eds.), *Improving Complex Systems Today*. London, Springer pp. 215-222.

Wartan, S. (2010). "Sharing Safety Knowledge for Aircraft Maintenance". Delft University of Technology, M.Sc. thesis.

Wiig, K. M. (1997). "Knowledge management: Where did it come from and where will it go?". *Expert Systems with Applications*, **13**(1): pp. 1-14.

Wognum, N. and A. Trappey (2008). "PLM challenges". *Advanced Engineering Informatics*, **22**(4): pp. 419-420.

Wood, R. E. (1986). "Task complexity: Definition of the construct". *Organizational Behavior and Human Decision Processes*, **37**(1): pp. 60-82.

Xue, D., S. Yadav and D. H. Norrie (1999). "Knowledge base and database representation for intelligent concurrent design". *CAD Computer Aided Design*, **31**(2): pp. 131-145.

Zein, S., B. Colson and S. Grihon (2012). "A primal-dual backtracking optimization method for blended composite structures". *Structural and Multidisciplinary Optimization*, **45**(5): pp. 669-680.

Appendix A: Complexity Estimation

The modular approach to development of KBS as expressed in the KLC ontology of Section 3.2 (e.g. reflected in the use of **Knowledge_Element** and **Process_Element** classes) poses a potential problem: the number of (potential) interfaces between these elements grows quickly, leading to increasingly complex systems. A similar issue is mentioned by Erden *et al.* (2008), who note that systems are evolvable if complexity does not increase in unmanageable amounts when new functionalities are introduced. Can this complexity be formulated?

The number of potential interactions between EKR elements can be either constrained or unconstrained. In the former case, constraints are applied to limit the number of element interactions: intra-class interactions are not allowed (e.g., a knowledge element cannot interact directly with another knowledge element). Figure A.1 shows the constrained case on the left-hand side.

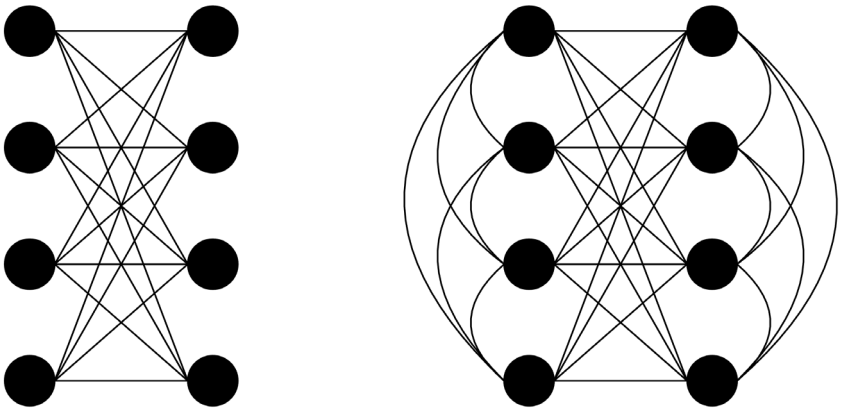

Figure A.1: Element interactions in constrained (left) and unconstrained (right) form

The constrained interaction form can be expressed using Equation 1:

$$g(n,m) = n \cdot m \qquad \text{(A-1)}$$

where *g* is the function expressing the total number of element interactions, *n* is the number of knowledge elements and *m* is the number of process elements.

In the unconstrained case, each knowledge or process element can interact with any other element. This is shown on the right-hand side of Figure A.1. This situation can be expressed using Equation A-2.:

$$h(n,m) = \sum_{i=1}^{n}(i-1) + \sum_{j=1}^{m}(j-1) + n \cdot m \qquad \text{(A-2)}$$

where *h* is the function expressing the total number of element interactions, without constraints. The number of resulting interactions for the unconstrained and constrained cases can be expressed by plotting functions *g* and *h* for any number of knowledge elements *n* and process elements *m.* In matrix notation, the following applies for these functions (where *h* can be substituted for *g*):

$$\begin{bmatrix} g(1,1) & \cdots & g(1,m) \\ \vdots & \ddots & \vdots \\ g(n,1) & \cdots & g(n,m) \end{bmatrix} \qquad \text{(A-3)}$$

Which results in the following plot for *g* and *h* (Figure A.2) when considering 10 knowledge and process elements (*n* = 1…10, *m* = 1..10). Figure A.3 shows the corresponding matrices.

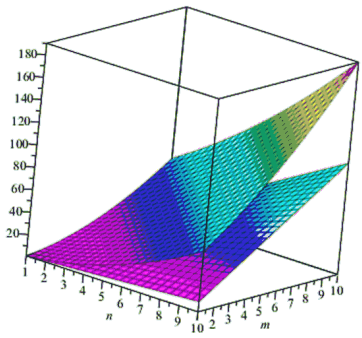

Figure A.2: Plot of element interactions for functions *g* and *h*, where *n*=1..10, *m*=1..10

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

1	3	6	10	15	21	28	36	45	55
3	6	10	15	21	28	36	45	55	66
6	10	15	21	28	36	45	55	66	78
10	15	21	28	36	45	55	66	78	91
15	21	28	36	45	55	66	78	91	105
21	28	36	45	55	66	78	91	105	120
28	36	45	55	66	78	91	105	120	136
36	45	55	66	78	91	105	120	136	153
45	55	66	78	91	105	120	136	153	171
55	66	78	91	105	120	136	153	171	190

Figure A.3: Number of element interactions for *g* and *h*, where *n* = 1..10, *m* = 1..10

Evidently, the total number of unconstrained interactions is dependent on the total number of elements; the mix is not important (e.g. when 8 elements are present, it can be 4 knowledge and 4 process, or 7 knowledge and 1 process – the total number of interactions are the same). However, when considering constrained interactions, the mix does make a difference.

Summarizing, the approach highlighted in this appendix allows for formulating an initial complexity estimate for modular system element interactions. The approach can be improved by a) grounding it more deeply in mathematical formulation; b) establishing a link with practice by researching hypothesized relations between the number of modular elements, system complexity and key performance indicators for implemented modular systems in industry.

Samenvatting

Een Ontologische Benadering voor Management van de Kenniscyclus binnen Fases van de Vliegtuiglevenscyclus

In het aircraft engineering domein zoeken producenten en operators constant naar verbetering van hun producten en processen. Knowledge-based applicaties worden in toenemende mate ontwikkeld om kennisintensieve engineering taken te ondersteunen of te automatiseren, wat leidt tot besparingen in tijd en geld.

Een primaire uitdaging hangt samen met het karakter en gedrag van kennis in de tijd. Verandert kennis en heeft het een levenscyclus? Huidig onderzoek met betrekking tot het onderwerp kennisverandering is echter vrij schaars. Verscheidene onderzoekers (bijvoorbeeld Schorlemmer et al. (2002), Alavi and Leidner (2001), Stokes (2001), Nonaka et al. (2000) en Schreiber et al. (1999)) geven aan dat kennis verandert, maar deze onderzoekers geven geen accurate definitie van hun concepten. Een aantal van hen geeft geen onderbouwing voor hun stellingen. Niemand gaat verder dan een kwalitatieve inschatting van kennisverandering.

Dit heeft aanzienlijke implicaties vanuit een praktisch perspectief. Bij verandering van kennis ontstaat een risico dat knowledge-based applicaties snel overbodig worden. Coenen and Bench-Capon (1993) geven een indicatie van de grootte van het probleem: het bestudeerde knowledge-based systeem was onderhevig aan 50% verandering van de bestaande regels per jaar, terwijl de gehele knowledge base met een factor vier groeide in de eerste drie jaar. Van Dijk *et al.* (2012) geven een indicatie van de kosten die onderhoud van een knowledge-based applicatie met zich meebrengt. Deze kosten worden geschat op 25% van de initiële investering *op jaarlijkse basis*.

Hoe kunnen knowledge-based applicaties dan omgaan met kennisverandering? Het is vereist dat modellen en methodes worden ontwikkeld voor het ontwerpen van meer robuuste engineering applicaties: bruikbaarheid en onderhoud van kennis en applicaties moet worden gefaciliteerd. Gezien deze overwegingen is het volgende algemene onderzoeksdoel gedefinieerd:

> *Ondersteuning van het consistent formaliseren, gebruiken en onderhouden van veranderende kennis binnen fases van de levenscyclus van vliegtuigen ter einde domein-specifiek modelleren, uitvoering en controle van engineering taken te verbeteren*

Kennisverandering wordt hier gedefinieerd als een verandering in kennis over tijd, waar kennis is gedefinieerd als verwerkte informatie met een capabiliteit voor

effectieve actie. De volgende types van verandering kunnen worden onderscheiden in een knowledge-based applicatie: verandering in waardes (**data change**), verandering in de gestructureerde context van een kennis-element (**information change**) en verandering in de capabiliteit voor effectieve actie geassocieerd met een kenniselement (**knowledge change**), wat kan worden veroorzaakt door veranderingen in regels, logische structuren of attribuut-sets.

Om het algemene onderzoeksdoel te realiseren zijn een aantal specifieke bijdrages aan de academische theorie ontwikkeld, wat tevens het beantwoorden van specifieke onderzoeksuitdagingen met zich meebracht – zie Tabel S.1.

Tabel S.1: Bijdrages aan theorie en geassocieerde onderzoeksuitdagingen

Onderzoeksbijdrage	**Geassocieerde uitdaging**
Knowledge Lifecycle Model	Karakteriseren, modelleren en kwantificeren van het gedrag van kennis binnen de levenscyclus van het vliegtuig
Ontologie-gebaseerde benadering voor het ondersteunen van kennisverandering: Knowledge Lifecycle ontology	Onderhoud: - 'black-box' KBS applicaties vermijden met behulp van transparantie Bruikbaarheid: - Taakoriëntatie - Betrokkenheid expert / eindgebruiker
Ontwikkeling van methodologie: KNOMAD methodologie	Methodologische benadering voor het faciliteren van kennisveranderingsmanagement

Het Knowledge Lifecycle Model is ontwikkeld om de levenscyclus van kenniselementen met behulp van twee concepten te karakteriseren en modelleren: kennis-staat en kennis-acties. De acties – creëren, formalizeren, gebruiken, onderhouden, updaten en archiveren – bieden de mogelijkheid om het gedrag van kennis in de tijd betekenisvol te kwantificeren. Deze mogelijkheid gaat voorbij de huidige state-of-the-art.

De ontwikkelde Knowledge Lifecycle (KLC) ontologie kan dienen als een structuur-behoudende benadering voor het ontwikkelen, gebruiken en onderhouden van knowledge-based applicaties.De KLC ontologie omvat twee centrale perspectieven: het Enterprise Knowledge Resource (EKR) concept gecombineerd met een annotatie-structuur gebaseerd op het Product-Process-Resource (PPR) paradigma. Een EKR is een taak-georienteerde representatie die kenniselementen, proceselementen en taak-output in de vorm van case rapportages bevat. In combinatie met het PPR paradigma kunnen 'white-box' knowledge-based applicaties worden ontwikkeld met verbeterde transparantie. De KLC ontologie gaat op vier manieren voorbij de huidige state-of-the-art: het maakt structuur-behoudend modelleren *en implementeren* mogelijk, het

representeert kennis in relatie met individuele taken, het biedt consistente annotatie via de PPR classes in relatie met individuele taken en het biedt systematische opslag van resultaten.

De derde en laatste bijdrage is de KNOMAD methodologie. Deze methodologie ondersteunt het ontwikkelen van knowledge-based applicaties die kunnen omgaan met kennisverandering. De methodologie bestaat uit zes stappen: Kennisacquisitie en Identificatie van Kennisverandering, Normalisatie, Organisatie, Modelleren en Implementatie, Analyse en Delivery. Het kritieke aspect van kennisverandering (en bijbehorend onderhoud) wordt ondervangen door het karakteriseren en analyseren van kennisverandering bij de start van het KNOMAD-proces. De organisatie-stap benadrukt het modelleren van het domein. Deze kennis kan in de volgende stap (Modelleren en Implementatie) worden gebruikt voor het annoteren van engineering taken. Het gebruik van de KLC ontologie wordt aanbevolen in de Modellering en Implementatie-stap. Derhalve worden domein en taakontologiën ontwikkeld en geïmplementeerd als de ruggegraat voor de ontwikkelde knowledge-based oplossingen. Als gevolg hiervan realiseren de verschillende KNOMAD-stappen een ontologie-gebaseerde benadering welke de onderzoeksuitdagingen met betrekking tot black-box applicaties en transparantie, taak-oriëntatie en betrokkenheid van de eindgebruiker/expert beantwoordt.

De ontologie-gebaseerde benadering en de geassocieerde modellen en methodologie zijn ter validering toegepast in drie case studies betreffende specifieke fases uit de vliegtuiglevenscyclus – ontwerp, productie en onderhoud.

Met betrekking tot de theoretisch georienteerde uitdaging - karakteriseren, modelleren en kwantificeren van het gedrag van kennis binnen een product levenscyclus – kan worden gesteld dat het Knowledge Lifecycle model succesvol is toegepast om kennisverandering in de ontwerp- en productiefases te karakteriseren. Het model is daarnaast succesvol toegepast in het onderhoudsdomein om kennisverandering te kwantificeren.

De uitdagingen met betrekking tot onderhoud – voorbij 'black-box' KBS applicaties bewegen en transparantie – zijn beantwoord door middel van de concepten van de KLC ontologie. Traceerbaarheid van kennis is bewerkstelligd via het EKR concept, met name door de Case class en de geassocieerde case rapporten. Deze maken het mogelijk om de uitkomsten van kennistoepassing voor een specifieke taak te traceren. Tevens kunnen de kennis en processes die zijn gebruikt voor uitoefening van de taak worden getraceerd. De metadata die is verbonden aan de kennis- en proceselementen (zoals auteurschap, levenscyclus-fase, status, etc.) helpen tevens met traceerbaarheid en betrouwbaarheid, validiteit en ownership van kennis. Het PPR paradigma helpt de zichtbaarheid van centrale concepten te bewerkstelligen. De drie case studies laten zien hoe een domein-specifieke extensie van de PPR classes semantische annotatie van

geimplementeerde EKRs mogelijk maakt. Dit maakt het makkelijker om knowledge-based applicaties en componenten te vinden, inspecteren en gebruiken. De uitdagingen met betrekking tot bruikbaarheid – taakoriëntatie en betrokkenheid van de expert / eindgebruiker – worden beantwoord door het EKR concept en het gebruik van een web-based kennismanagement systeem. In elke case study is een of meer EKRs ontwikkeld voor het representeren en uitvoeren van specifieke engineering taken. De gekozen web-based architectuur faciliteert interactie van de gebruiker met EKRs en de bijbehorende componenten (kenniselementen, proceselementen, cases).

De KNOMAD methodologie is in alle case studies toegepast. Alle stappen van de methodologie zijn succesvol toegepast voor het ontwikkelen en implementeren van knowledge-based applicaties die met kennisveranderingen kunnen omgaan.

De bijdrages van deze dissertatie – Knowledge Lifecycle model, KLC ontologie, KNOMAD methodologie – kunnen op verscheidene manieren worden uitgebreid en verbeterd. Voor het Knowledge Lifecycle is met name van belang dat het kwantitatief wordt gevalideerd voor meerdere domeinen. Ideaal gesproken wordt dit model ook voorzien van een formele mathematische onderbouwing. Daarnaast is taakcomplexiteit en –hierarchie niet gemodelleerd en bestudeerd in deze dissertatie. Als laatste moet worden genoteerd dat de KLC ontologie kan worden uitgebreid met formele expressies, wat het gebruik van reasoning technieken in het ontwerp en onderhoud van knowledge-based applicaties zou faciliteren.

Curriculum Vitae

Wilhelmus (Wim) Johannes Cornelis Verhagen was born in Moergestel, The Netherlands on 29 September 1984. Wim attended high school at the Onze Lieve Vrouwe Lyceum in Breda, graduating cum laude in 2002. He subsequently elected to pursue a degree in Aerospace Engineering at Delft University of Technology. Wim completed his B.Sc. in 2006 and joined the erstwhile department of Aerospace Management & Operations (AMO), attaining his M.Sc. (cum laude) in 2008 in the renewed Air Transport & Operations (ATO) chair under supervision of prof. Curran and dr. Beelaerts van Blokland.

Following his graduation, Wim was pointed towards an opportunity to research manufacturing knowledge management in cooperation with Airbus UK, and later EADS Innovation Works. This initial project grew into his Ph.D. research, under supervision of Prof. Curran, where a life-cycle focus (both for knowledge and for aircraft) was determined as research progressed. Over the past four years, Wim (co-)authored four journal publications and four conference papers directly related to the topic of his dissertation, besides authoring various other publications. Wim has also reviewed for such journals as Advanced Engineering Informatics, Expert Systems with Applications and the Journal of Systems Science and Systems Engineering. He continues to explore the various fields related to the research domains addressed in the dissertation. During Ph.D. employment, Wim supervised 4 M.Sc. students and lectured in the faculty's third year Systems Engineering course. He gladly supported the international activities of the ATO group through involvement in the International Society of Productivity Enhancement (ISPE) and the Air Transport and Operations Symposium series.

Outside of his immediate research exploits, Wim takes an interest in research in general, ranging from research methods to specific domain research, ranging from evolutionary biology to astronomy, from history to economics. His hobbies, outside of stuffing his nose into books and papers, are sports such as running, mountainbiking and squash, and outdoors activities such as mountain hiking and exploring countries. Since a few years, Wim is an active supporter of cancer research in the Netherlands, participating in Alpe d'HuZes 2012 and Relay for Life 2013.

List of Publications

Bermell-Garcia, P., W. J. C. Verhagen, S. Astwood, K. Krishnamurthy, J. L. Johnson, D. Ruiz, . . . R. Curran (2012). "A framework for management of Knowledge-Based Engineering applications as software services: Enabling personalization and codification". *Advanced Engineering Informatics*, 26(2): pp. 219-230.

Verhagen, W. J. C., P. Bermell-Garcia, R. E. C. Van Dijk and R. Curran (2012). "A critical review of Knowledge-Based Engineering: An identification of research challenges". *Advanced Engineering Informatics*, 26(1): pp. 5-15.

Verhagen, W. J. C., P. Bermell-Garcia, P. Mariot, J.-P. Cotton, D. Ruiz, R. Redon and R. Curran (2012). "Knowledge-based cost modelling of composite wing structures". *International Journal of Computer Integrated Manufacturing*, 25(4-5): pp. 368-383.

Curran, R., W. J. C. Verhagen, M. J. L. Van Tooren and A. H. Van der Laan (2010). "A multidisciplinary implementation methodology for knowledge based engineering: KNOMAD". *Expert Systems with Applications*, 37(11): pp. 7336-7350.

Verhagen, W. J. C. and R. Curran (2013). "A Knowledge Lifecycle Model for Measurement of Knowledge Change". J. Stjepandić, G. Rock and C. Bil (eds.), *Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment*. London, Springer, pp. 279-290.

Verhagen, W. J. C. and R. Curran (2011). "Ontological Modelling of the Aerospace Composite Manufacturing Domain". D. D. Frey, S. Fukuda and G. Rock (eds.), *Improving Complex Systems Today*. London, Springer pp. 215-222.

Verhagen, W. J. C. and R. Curran (2010). "Knowledge-Based Engineering Review: Conceptual Foundations and Research Issues". J. Pokojski, S. Fukuda and J. Salwiński (eds.), *New World Situation: New Directions in Concurrent Engineering*, London, Springer, pp. 267-276.

Curran, R., W. J. C. Verhagen and M. J. L. Van Tooren (2010). "The KNOMAD methodology for integration of multi-disciplinary engineering knowledge within aerospace production". *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, Florida.

Curran, R., W. J. C. Verhagen, A. H. Van der Laan and M. J. L. Van Tooren (2009). "KBE and Manufacturing Constraints Management". S.-Y. Chou, A. Trappey, J. Pokojski and S. Smith (eds.), *Global Perspective for Competitive Enterprise, Economy and Ecology*, Springer London, pp. 783-791.